MANGALORE **UNIVERSITY**

A Research Proposal

[Pre-Thesis Synopsis]

Submitted to carry out research leading to a Ph.D. degree in Computer Science

Title of the Work

Performance Analysis of Secure Smart Home System Based on IoT

BY

SUPARNA N

Research Scholar

Reg. Nor: MU/EXB/Ph.D./CR.40/Comp.sc./2018-19/E.25

Under the Guidance of

Dr. Manjaiah D H

Senior Professor and Chairman

DEPARTMENT OF POST-GRADUATE STUDIES AND RESEARCH IN COMPUTER SCIENCE

MANGALORE UNIVERSITY

MANGALAGANGOTHRI-574 199

KARNATAKA, INDIA

TABLE OF CONTENTS

Sl.No.	Description		Page No.
1	INTRODUCTION		7
	1.1	Smart Homes Past, Present and Future	4
	1.2	Smart Home Infrastructure	5
2	PROBLEM STATEMENT		6
3	LITERATURE REVIEW		6
4	OBJECTIVES		
	4.1	Implementation of LWC algorithm- IDEA	9
	4.2	Implementation of LWC algorithm- PRESENT	15
	4.3	Implementation of LWC algorithm- HIGHT	23
	4.4	Detection and Mitigation of DDoS Attack in SHNs	26
5	CONCLUSION		48
6	FUTURE SCOPE		48
	REFERENCES		49
	PUBLICATIONS		51

Performance Analysis of Secure Smart Home System Based on IoT

1 INTRODUCTION

A home is more than just a place to live. It involves us, our people and the things we love and value the most. With smart home solutions we can control and showcase our home elegantly with single touch as we keep ourselves engaged in various day to day activities. We can take control of the household applications on our mobile phone.

Our home is a cradle which gives us, the relaxation and happiness. So, we indulge in buying items which provide comfort, easiness and luxury, even though some products are expensive. But in the pursuit of luxury, can we forget or forgo safety? When we are not at home, some irksome little doubts start filling our mind. “Did I turn off coffee maker?”, “Did I set the security alarm?”, Did I Switch off the geyser?” “Are the kids doing homework or watching TV?” etc. With a smart home, we can quite all these worries with a quick glance on our mobile phone or laptop. We can connect the devices and appliances that we use daily at home, to each other and with us. A sensor can be attached to any device in a home and can be put on a home network with an internet connection, thus gives the facility of controlling the entire house by owner’s commands. A smart home is defined in [1] as “A smart home, or smart house, is a home that incorporates advanced automation systems to provide the inhabitants with sophisticated monitoring and control over the building’s functions. For example, a smart home may control lighting, temperature, multi-media, security, window and door operations, as well as many other functions.”

Smart homes make life more convenient and safer. Though a smart home set up gives the impression of luxury, the lifestyle of present generation people has made it a necessity. Whether we are at work or on vacation, the smart home will alert us to what is going on inside and around our house such that we can build a security system to provide an immense amount of help in an emergency. From adjusting the thermostat to controlling the house or garage doors with a simple voice command, our home becomes a personalized asylum.

1.1 Smart Homes Past, Present and Future

Smart homes are defined as the environments designed to assist individuals with daily domestic activities using automated machines or internet-enabled devices. They can be classified according to the types of equipment and systems installed. The major targets are improving comfort, dealing with medical rehabilitation, monitoring mobility and physiological parameters, and delivering therapy. Technologies exist to help people deal with a reduction or loss of mobility, vision, hearing, and cognitive ability; to continuously monitor vital parameters; to reduce accidents by anticipating risky situations; and to deliver therapy through wearable biomedical sensors. All these systems maintain a certain level of independence, thus providing a better quality of life for the residents and their close relatives. However, the residents of smart homes are not just those with severe pathologies or chronic illness; there are also those who simply want a better quality of life[2].

1.2 Smart Home Infrastructure

A smart home is a living environment equipped with technology that enables the automated and remote control of all household devices and appliances. In

this setup, users can comfortably monitor and control their home devices from a mobile phone or a laptop via the internet. Home appliances are integrated within a well-defined network architecture using standardized protocols for communication. The complete arrangement can be generally categorized into two layers. The first, Home Automation Layer encompasses all the home devices, including switches, modules, RF transmitters/receivers, actuators and sensors. The second, Home Management Layer comprises interface devices, processors, data collectors, and gateways that facilitate communication with the internet[3].

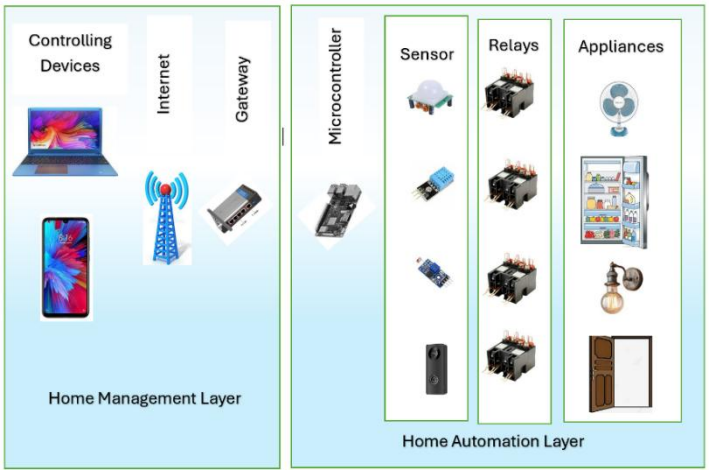


Figure 1.1: Smart Home Infrastructure

Figure 1.2 illustrates the overall control function of the Smart Home Network system. We have created a functional smart home network with a range of devices and capabilities, using Cisco Packet Tracer. This set up has facilities like remote control, automation, monitoring, and enhanced security in a connected home environment.

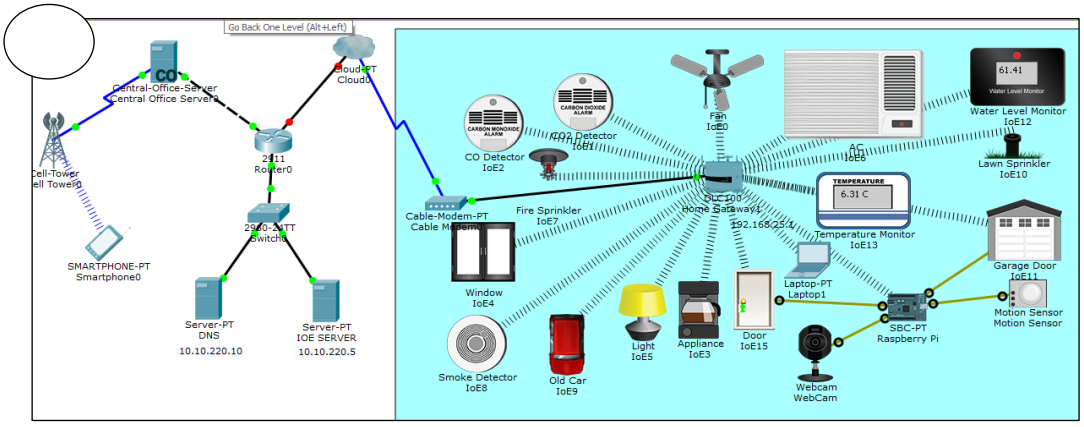


Figure 1.2: Completely Automated Smart Home

This infrastructure includes network components, smart home management devices, data acquisition devices (sensors), and both wired and wireless connections. The network infrastructure primarily consists of a switch,

router, cable modem, central server, cell tower, and cloud account. The smart home devices can be categorized into two types: wired devices and wireless devices. Wired devices include a DNS server and a laptop, while wireless devices consist of a smartphone used for remote control and monitoring of the smart home, along with various sensors such as smoke detectors, CO detectors, motion sensors, door sensors, window sensors, temperature sensors, water level monitors, humidity sensors, light sensors, and the Raspberry Pi microcontroller. The connectivity in this infrastructure is mainly of three types: wired, wireless, and cellular connections.

In this setup, there are three major functionalities of the smart home. Users can control appliances, lights, and other devices remotely using a smartphone app. The system can automate tasks based on sensor readings (e.g., turning on lights when motion is detected or activating sprinklers based on soil moisture levels). Users can monitor the status of sensors (smoke, CO, temperature, and water level) and receive alerts. Additionally, webcams and motion sensors can enhance home security, while smart control of appliances and lights can help optimize energy consumption.

2 PROBLEM STATEMENT

The IoT-based smart home networks consist of resource constrained devices and therefore are highly vulnerable to various types of attacks in each layer of their architecture. Hence, we need to set up a state-of-the-art security mechanism either using cryptography or using protocol-based detection and mitigation techniques.

To prevent malevolent attempts at the application layer a Lightweight IDS (LWIDS) which implements the Light-weight cryptography algorithms IDEA,HIGHT and PRESENT at the application layer is designed and developed in this project. Secondly to prevent malicious attacks such as DDoS at the network layer, a prevention and mitigation mechanism for DDoS attack named AtRouterIDS(ARIDS) is implemented. The LWIDS prevents all kinds of malicious activities by encrypting the messages from sensors to Gateways and ARIDS detects and prevents DDoS attack at Network layer with RPL protocol.

3 LITERATURE REVIEW

The smart home networks are vulnerable to various kinds of security threats and attacks as they are highly resource constrained systems. We conducted an elaborative survey of wide range of articles which are listed in the table 3.1 below:

Paper and Authors	Title of the Paper	Journal and Publication Year	Area of study/Type of Study	Methodology	Outcome of the Research Work
[4] Fraser Hall et al 2010	Smart Homes: Security Challenges and Privacy Concerns	arXiv: 2010.15394v1	Collation of information from various sources	Article reference	Presented a review of core technologies and applications that define the smart home, and collated the primary security and privacy concerns faced by smart homes
[5] Sahabul Alam and Debashis De 2014	Analysis of Security Threats in Wireless Sensor Network	International Journal of Wireless & Mobile Networks	Collation of information from various sources	Article reference	Presented a comprehensive study of threats on wireless sensor networks
[6] Gaurav Sharma et al 2021	A Survey on Layer-Wise Security Attacks in IoT: Attacks, Countermeasures, and Open-Issues	Electronics 2021, 10, 2365	A review study and study of performance analysis of IoT network with implementation of different attacks	Simulation of different types of attacks using Cooja Simulator	Reviewed the layer-wise security protocols for WSN and IoT. Authors identified some novel research opportunities in the security issues of both WSN and IoT. The work also tested the performance analysis of IoT network under several attacks using Cooja Simulator.
[7] Hamza Sajjad et al 2019	Evaluating Security Threats for each Layers of IoT System	Research Gate	Review of articles	Article Reference	This paper presents potential attacks on each layer of IoT and explains their impact on the network. Security measures, along with possible solutions for these networks, are also discussed.
[8] Joseph Bugeja et al 2016	On Privacy and Security Challenges in Smart Connected Homes	European Intelligence and Security Informatics Conference	Survey of research articles	Article Reference	Researchers provided an overview of the privacy and security challenges in the smart home domain. They identified constraints, evaluated potential solutions, and discussed various challenges and research issues that need further investigation. Furthermore, they outlined mitigation approaches applicable at different architectural levels.

[9] Ayush Gai et al 2018	Categorisation of security threats for smart home appliances	International Conference on Computer Communication and Informatics	Survey of research articles	Article Reference	The writers concentrated on identifying and categorizing threats while creating threat vectors for IoT-based home appliances. They selected several home appliances for testing and analyzed them to uncover attack surfaces and vulnerabilities. Their research revealed over 15 attack surfaces and 11 vulnerabilities across these devices.
[10] Izzat Alsmadi et al 2019	IoT security threats analysis based on components, layers and devices	American Journal of Science and Engineering	Literature review	Literature review	The authors assessed security threats and attacks on IoT networks by categorizing them into three main areas: IoT vulnerabilities, threats and attacks categorized by the layers of IoT architecture, and vulnerabilities, threats, and attacks associated with specific IoT components..
[11] Alberto E Coboi et al 2021	Security Problems in Smart Homes	ICSES Transactions on Computer Networks and Communications	Literature review	Literature review	The authors prescribed new security techniques for smart homes technologies such as Machine-learning algorithms, Real time algorithms, Deep learning algorithms, Prediction algorithms, Artificial Neural Network algorithms, Blockchain to security and Identity-based encryption
[12] Seokung Yoon et al 2015	Security Issues on Smarthome in IoT Environment	Springer-Verlag Berlin Heidelberg Lecture Notes in Electrical Engineering 330	Literature review	Literature review	This paper analyses the possible attack scenarios in smart home and proposes countermeasures.

Table 3.1 Literature Review about Security Issues in IoT-based Smart Home Networks

4 OBJECTIVES

4.1 IMPLEMENTATION OF LWC ALGORITHM- IDEA

4.1.1 Introduction

International Data Encryption Algorithm, or IDEA for short, was originally known as IPES (Improved Proposed Encryption Algorithm) and was developed by Xuejia Lai and James L. Massey in the year 1991. The authors presented the algorithm in their paper [2]. The algorithm follows the concept of "Mixing operations from different algebraic groups." The cipher structure provided both confusion and diffusion, making it possible to implement using hardware as well as software. Three distinct group operations were successively performed on pairs of 16-bit sub-blocks that provide the required confusion, and the ciphered text was chosen for necessary diffusion. IDEA encrypts 64-bit plaintext to a 64-bit ciphertext using a 128-bit length key. IDEA uses a complicated Mangler function that does not require reversing the steps of encryption to do decryption. All the steps of encryption and decryption are identical except the key expansion procedure.

4.1.2 The Basic Structure

IDEA takes 64-bit plain text as input and breaks it to smaller chunks or sub-blocks of 16 bits each. These sub-blocks are passed onto first round. IDEA consists of 17 rounds.

Problem Statement

Although various types of research have been done to detect and mitigate DDoS attack in IOT and WSNs, a study with ContikiRPL is seldom done. The IoT networks consists of resource constrained devices which use tiny operating systems like TinyOS, Contiki etc. There are many unresolved security concerns in Contiki based IoT networks and the development of a benchmark Intrusion Detection System is still far behind. In this project we have implemented an IDS which works inside the topology of a IoT network which secures the communication between the sensor devises. This study analyses the performance metrics such as PDR, Power Consumption, Latency etc., setting up different frameworks and test cases. The performance of the network with and without IDS is also compared.

The 128-bit key is given to a key expansion process which will generate 52 sub-keys of 16 bits each viz., k_1, k_2 , k_3, ……, k_{52}. IDEA is stronger than other lightweight encryption algorithms because it uses 52 subkeys and it undergoes 17 rounds of encryption process. In the round 1 two 16 bit operands are taken and three primitive operations are performed. The two operands are 16-bit plain text and a 16-bit key. The primitive operations are Bitwise XOR, slightly modified binary addition (+) and slightly modified binary multiplication. The binary addition of two 16-bit operands will lead to a 16 or 17-bit resultant. But for encryption purpose only 16-bit resultant is required. Therefore, the binary addition is slightly modified by discarding the carry bit generated during addition. Similarly, the multiplication of two 16-bit numbers will generate a 32-bit number. The multiplication operation is slightly

modified by dividing the resultant number by $2^{16}+1$ and the remainder is considered for the further process.

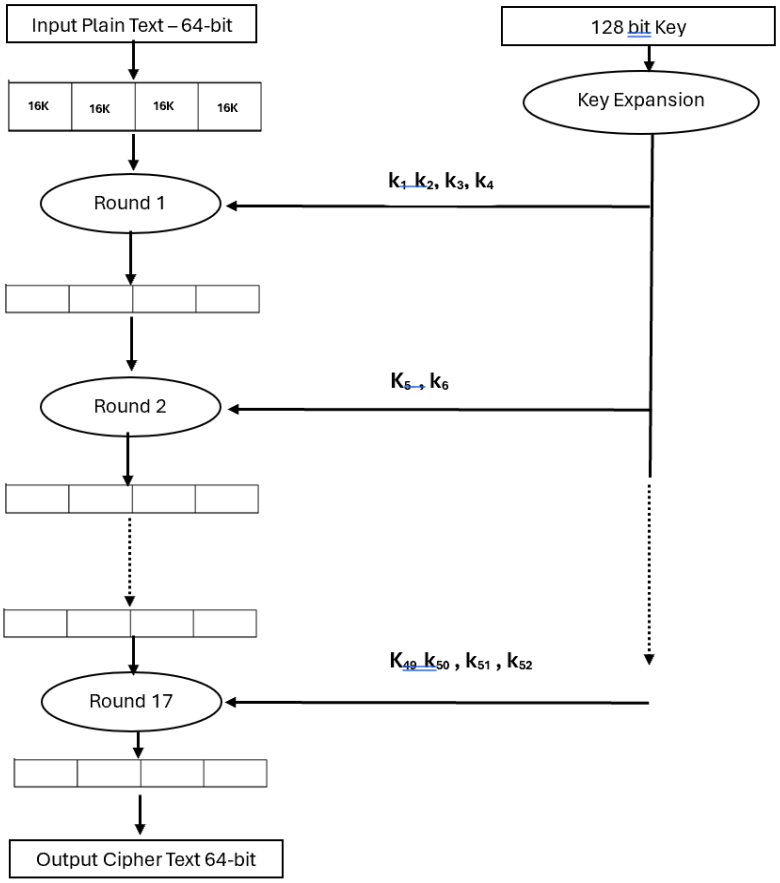


Figure 4.1: Basic Structure of IDEA algorithm ([13]

4.1.3 Motivation and Contribution

To secure smart home network, the lightweight cryptographic algorithm IDEA is implemented in SHN using the Cooja Simulator in Instant Contiki OS. This work is published in IEEE XPLORE[14].

4.1.4 Review of Literature

Various researchers have implemented IDEA in Software as well as in Hardware. The developers of IDEA viz., Xuejia Lai and James Massey used the design concept of "mixing operations from different algebraic groups". The required confusion is achieved by successively using three different group operations on pairs of 16-bit sub-blocks and the cipher structure was chosen to provide the necessary diffusion. The cipher is so constructed that the deciphering process is the same as the enciphering process once the decryption key sub-blocks have been computed from the encryption key sub-blocks[13].

Three different group operations are successively used on pairs of 16-bit blocks to achieve confusion and the cipher structure provided essential diffusion. The construction of cipher was such that the encryption and

decryption processes remained the same once the decryption key sub-blocks have been computed from the encryption key sub-blocks. The regular modular structure used for the encryption process ensured the easy implementation of the algorithm in hardware[13].

Xuejia Lai and James Massey [13] implemented IDEA using VLSI at the Integrated Systems Laboratory of the ETH, Zurich. The estimated data rate of this chip varies from 45 Mbits per second to 115 Mbits per second, depending on the complexity of the architecture chosen

In 1993, the six researchers in ETH Zurich namely R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber, and W. Fichtner from Integrated Systems Laboratory of the ETH, Zurich implemented a more secure IDEA crypto algorithm with online failure detection mechanism. This implementation produced a high data throughput of 177Mbits/s with a system clock frequency 25MHz. Here four full custom modulo $(2^{16}+1)$ multipliers[15] are used.

We can find several types of implementations, improvisations, and optimizations of IDEA in hardware, but software implementations are sparse. However, with the emergence of various simulation tools for network implementations, research in Wireless Sensor Networks and the Internet of Things has increasingly relied on these tools. An extensive list of simulators, such as MATLAB, NS2, NS3, OMNET++, NETSIM, Cooja, TOSSIM, Cisco Packet Tracer, and OPNET, has enabled numerous researchers to implement and analyze cryptography algorithms. In this paper, we have implemented IDEA in Smart Home Networks using the Cooja Simulator in Contiki OS.

Paper	Method of Implementation	Encryption Rate	Year
[16] VLSI Implementation of IDEA Encryption Algorithm	Verilog VHDL Implementation	-	1992
[17]A VLSI implementation of RSA and IDEA encryption engine.	VLSI Implementations XILINX FPGA	-	1998
[18]A Bit Serial Implementation of the International Data Encryption Algorithm IDEA	Xilinx Virtex XCV300 FPGA	500Mb/sec	2000
[15] VINCI: VLSI implementation of the new secret-key block cipher IDEA	1.2$^{\mu m}$ CMOS technology	177Mbps	2004
[19]VLSI Implementation of a New Block Cipher	1.5$^{\mu m}$ CMOS technology	encryption rate of 44Mbps	1992
[20] On the VLSI implementation of the international data encryption algorithm IDEA	0.8$^{\mu m}$ CMOS technology	355Mbps	1995

Table 4.1: Related works on Hardware Implementations of IDEA

4.1.5 Communication Protocols

Nowadays, different communication protocols have emerged with the availability of various resources. Communication protocols enable reliable and uninterrupted data transmission. Some of the well-known communication protocols for IoT, include MQTT, CoAP, RPL, VSCP, and

XMPP. We have implemented this project using MQTT due to its adaptability for this particular purpose.

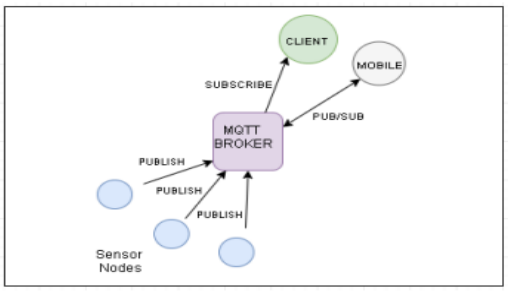


Figure 4.1: MQTT Architecture

Message Queuing Telemetry Transport (MQTT) was developed in 1999 by Dr. Andy Stanford-Clark of IBM and Arlen Nipper of Arcom. This protocol is based on a publish-subscribe model and employs a client/server architecture, where each sensor acts as a client connecting to a server known as a broker over TCP. As a message-oriented protocol, MQTT treats each message as a distinct piece of data, which remains unknown to the broker. Messages are published to specific addresses called topics. An MQTT client should consistently maintain a connection to the broker. Although MQTT is lightweight, maintaining an uninterrupted TCP connection can be challenging when implemented in constrained devices that experience high packet loss and have limited computing resources. Topic names in MQTT are often long strings and incompatible for 802.15.4. So we have used MQTT-SN protocol, which defines a UDP mapping of MQTT and adds broker support for indexing topic names[21].

4.1.6 Simulation

IoT-based wireless networks, such as Smart Homes, Smart Cities, and Smart Grids, utilize devices with constrained resources, including sensors, RFID tags, routers, and gateways. Several tiny operating systems, both proprietary and open source, are available for IoT; among them, Contiki stands out as a renowned lightweight operating system based on the C programming language. The performance analysis of IoT-based smart home networks can be effectively studied by simulating the networks using the Cooja network simulator in Contiki. In this project, we analyzed the performance of smart home networks with 15 nodes and 30 nodes, both with and without security features. Contiki includes built-in codes in C and operates on both IPv4 and IPv6.However, in an IoT network, data communication between sensor nodes and gateways remains unencrypted, exposing the network to several security threats, including DDoS attacks. To enhance security, it is essential to deploy cryptographic algorithms in the data paths between servers and clients or between sensors and gateways. Lightweight cryptographic algorithms developed in C, such as PRESENT, IDEA, HIGHT, and TEA, can be easily implemented in Cooja, providing dual-layer security for smart home networks. To analyse the performance metrics, we need to consider the number of nodes, the number of interfaces for each node, the number of simulation observers,

and the simulation duration. We simulated two separate networks, consisting of 15 and 30 native code nodes, with three types of nodes: brokers, publish nodes, and subscribe nodes. The hardware configuration for the simulation is crucial in Cooja, and we designed both networks using Z1 motes, a general-purpose wireless sensor platform compatible with the TMote family[22]. The Z1 mote offers features that make it an ideal choice for smart home network implementations.

1 A second generation MSP430F2617 low power microcontroller.
2. A powerful 16-bit RISC CPU @16MHz clock speed
3. 8KB RAM and a 92KB Flash memory.
4. CC2420 transceiver.
5. It operates at 2.4GHz with an effective data rate of 250Kbps
6. Many sensors can be connected to the Z1 mote.
7. I2C sensors can be connected to the Ziglet defacto port.
8. It does not require additional HW to program.
9 It can be powered with batteries (3.3V) or using 5V from the u-USB port.
10. It guarantees the maximum efficiency and robustness with low energy cost.[23].

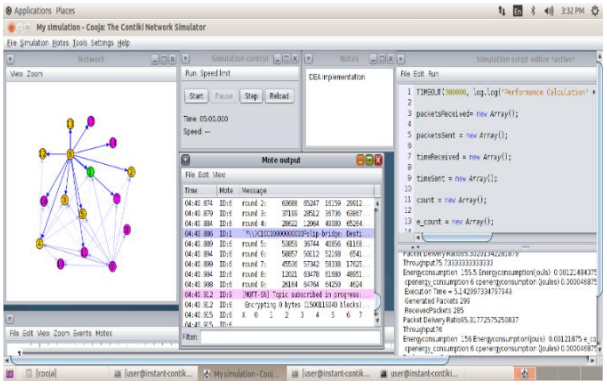


Figure 4.2: Simulation of Smart Home Networks in Cooja

4.1.7 Performance Analysis

Smart Home Networks with 15 nodes and 30 nodes are deployed using Z1 mote and the performance metrics such as Throughput, Packet Delivery Ratio, Execution Time and Energy Consumption are observed from repeated simulations. The average values of these parameters are listed in the table below. At the second stage, IDEA cryptography algorithm is implemented in the Network and the results are observed and tabulated in Table4.2. The comparison of performance metrics with and without IDEA is depicted in the graph at Figure 4.3 and 4.4.

Performance	With IDEA		Without IDEA	
	15 Nodes	30 Nodes	15 Nodes	30 Nodes
Throughput (bps)	74.4	97.6	85.35	98.2
PDR	95.2	91.8	96.45	94.35
Delay(ms)	45.01	22.04	15.25	14.29
Energy Consumption (joules)	0.00124	0.00161	0.00034	0.00059
Execution Time(s)	5.3	3.4	3.5	4.2

Table 4.2 : Performance Metrics in Smart Home Networks

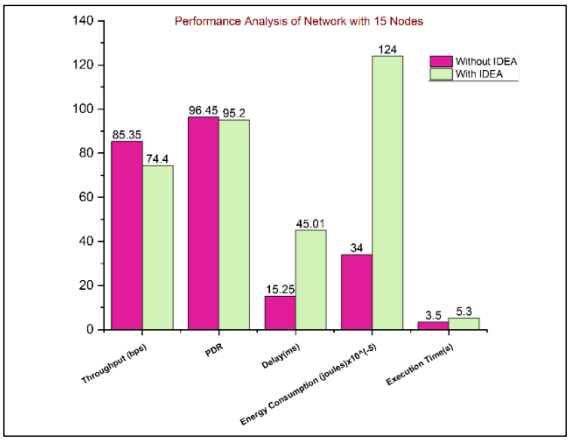


Figure 4.3: Performance Analysis of network with 15 nodes

Metrics and Their Meaning:

- Throughput (bps): A network's throughput measures the amount of data successfully transmitted per second (bits per second) across the nodes of the network. The higher the throughput, the better the network's performance.
- PDR (Packet Delivery Ratio): The percentage of packets that reach their destination successfully.

  *PDR = (Total number of packets received) / (Total number of packets sent) * 100*
- Delay (ms): The time taken by a packet to travel from the source to the destination, measured in milliseconds. Lower delay is better, indicating faster communication.
- Energy Consumption (joules): The energy used by thenodes to process and route data packets in the network. Since IoT devices operate on small batteries, lower energy consumption is preferable for longer battery life.
- Execution Time (s): The time taken to complete a specific task or operation, measured in seconds. A lower execution time indicates better network performance.

The implementation of the IDEA lightweight cryptography algorithm in this 15-node IoT network results in:

- Throughput measured with IDEA is lower compared to that measured without IDEA. This is mainly due to the processing time required for the encryption and decryption processes, which delays the transfer of data packets.
- The packet delivery ratio (PDR) shows an insignificant increase with the implementation of IDEA in the network.
- The increase in delay for data transfer in the network raises latency, thus affecting the network efficiency.
- The substantial increase in energy consumption leads to a considerable reduction in battery life.
- The significantly increased execution time also negatively impact the network's performance.

The implementation of the IDEA lightweight cryptography algorithm in the 30-node IoT network also has shown similar behaviour with some enhanced metric measurements because of the depth of the network.

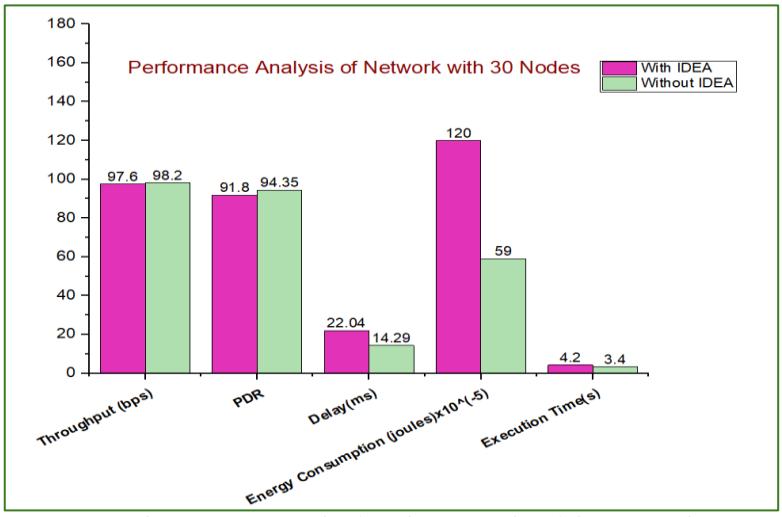


Figure 4.4: Performance Analysis of Network with 30 Nodes

4.1.8 Conclusion

The project demonstrates the performance implications of including cryptographic security features to an IoT-based smart home network and the study compares the consequences before and after the implementation. As IDEA offers tremendous security benefits, the trade-offs in terms of throughput, energy consumption and execution time need to be taken care of depending on the certain requirements of the smart home networks. It is vital to contemplate that these results are specific to this network set up which specially designed using Instant Contiki operating system. Different operating systems, hardware, topology or traffic pattern would yield different results. This project can be implemented directly on hardware devices which can be used in real life design of smart homes.

4.2 IMPLEMENTATION OF LWC ALGORITHM- PRESENT

4.2.1 Introduction

The PRESENT algorithm is a symmetric encryption technique characterized by its minimalist design, utilizing 4-bit substitution and permutation blocks as its foundation. This compact structure enables implementation on devices with limited hardware resources, making it well-suited for processors with narrow buses and efficient resource utilization. Compared to similar algorithms, PRESENT features relatively short passwords and requires fewer rounds. Its design and implementation pose an intriguing challenge, demanding a careful balance between resource consumption and overall system communication speed. This approach makes the algorithm particularly appealing for applications like smart home networks based on IOT, where hardware constraints are a significant consideration.

4.2.2 Encryption Process

PRESENT is a compact block cipher with several key components. It utilizes a 64-bit state derived from a plaintext input. The cipher supports two key

sizes: 80-bit and 128-bit. The algorithm includes a substitution box layer (S-Box) followed by a permutation layer (P-Layer). Additionally, a key register updates the key during each iteration of the algorithm. PRESENT operates through an iterative process, where the plaintext undergoes multiple rounds of processing through the S-Box, P-Layer, and key register before generating the final ciphertext. This process can involve up to 32 rounds of computation to produce the encrypted output[24]. The framework of the algorithm is represented in the below figure 4.5.

- All block ciphers contain a sequence of operations that are equivalent to ByteSub, ShiftRow, MixColumn, and AddRoundKey. This operational sequence, termed a round, must include processes based on uniform and invertible transformations, known as layers. These layers are purposefully designed to resist linear and differential cryptanalysis. The layers are described as follows:
- Non-linear layer or Substitution Layer: This layer applies S-boxes in parallel, exhibiting optimal non-linear properties.
- Linear mixing layer or Permutation Layer: This layer ensures a high level of diffusion across the multiple rounds.
- Key addition layer: This layer involves a unique OR operation between the intermediate stage and the specific key for each round and adds the key to the plain text[25].

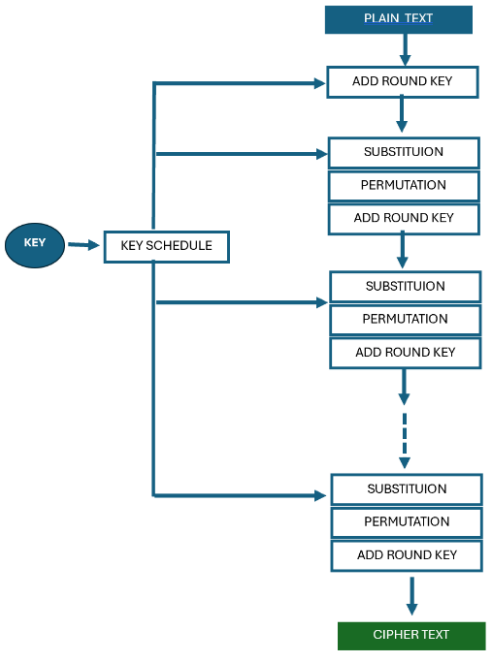


Figure 4.5: The general structure of PRESENT algorithm encryption process

4.2.3 Motivation and Contribution

To secure smart home network, the light weight cryptography algorithm PRESENT is implemented in SHN using the Cooja Simulator in Contiki OS.

4.2.4 Review of Literature

PRESENT is a Lightweight cipher developed in 2007 by Orange Labs in France, Ruhr University Germany and Technical University of Denmark. It is popular for its tiny size which is about 2.5 times smaller than AES. PRESENT is recommended for use in low-power environments and high-performance applications such as smart home networks, smart grids and smart city environments. The ISO and the International technical Commission incorporated PRESENT as the new International Standard for LWC methods[26].

Present algorithm was developed by A. Bogdanov et al and the design, implementation and cryptanalysis was explained in their paper PRESENT: An Ultra-Lightweight Block Cipher[26]. Further in 2017 Cihangir Tezcan et al conducted differential cryptanalysis of PRESENT algorithm and presented in their paper "Differential Attacks on Lightweight Block Ciphers PRESENT, PRIDE, and RECTANGLE Revisited" which was published in the Springer LNCS Lightweight Cryptography for Security and Privacy, compiled by Andrey Bogdanov[27].

The 80-bit key version of PRESENT requires approximately 1,600 logic gates, and the encryption of 64-bit plaintext runs for 32 clock cycles. Typical IoT devices operate at a clock rate of around 1 MHz. A pipelined implementation of PRESENT with 31 encryption stages achieves a throughput of 64 bits per clock cycle, resulting in a total throughput of over 50 Gbit/s[28].

Daniel Irwin et al in their work [24], implemented PRESENT algorithm in both hardware and software, and compared its performance on different platforms. The authors used an Intel I5 Laptop with Windows 8, 8GB Ram and 2.4GHz speed, A BeagleBone Black microcontroller with Lightweight OS Linux Debian, ARM CORTEX A8 processor which had 512MB RAM and a ZedBoard microcontroller with Custom Linux, ARM CORTEX A9 processor which had 256KB RAM. They also implemented the C and Python implementations of the PRESENT encryption and Decryption processes. The researchers claimed that hardware implementations are multitudes better than the software implementations.

In their study "Design Space Exploration of Present Implementations for FPGAs" [29], Mohamad Sbeiti et al, investigated the efficacy of the PRESENT block cipher across various FPGA configurations. They implemented encryption and decryption functions in VHDL for the Spartan-III XC3S400 FPGA. The team utilized Mentor Graphics ModelSim XE 6.2g for simulations and Xilinx ISE v10.1.03 WebPack for design synthesis. Their analysis encompassed major cryptographic algorithms, including PRESENT, AES, ICEBERG, and SEA, focusing on FPGA implementation and performance evaluation. The researchers obtained performance metrics for PRESENT's encryption and decryption of a single data block, considering different key sizes and S-Box implementation methods. Their findings indicated that PRESENT is particularly suitable for applications requiring high speed and

high throughput, demonstrating notable hardware efficiency, especially in terms of throughput per slice.

In a separate study [30], Hengameh Delfan Azari and Prashant V. Joshi executed both 80-bit and 128-bit key versions of the PRESENT algorithm on an FPGA-based architecture. They employed Verilog-HDL on the Xilinx 14.7 ISE platform, using an Artix-7 FPGA board. Their analysis considered performance parameters such as slices, LUTs, flip-flops, frequency, and throughput. The researchers then compared their results with those reported in [20].

William Diehl et al, implemented five lightweight algorithms namely SIMON, SPECK, PRESENT, LED and TWINE in hardware using RTL (Register Transfer Level) and in software using the custom reconfigurable processor and compared the results obtained for these algorithms with that of AES, implementing AES in the same configuration. For the hardware implementations Xilinx Kintex-7 FPGA was used and parameters like throughput, area, throughput to area ratio, power and energy were noted down for each of these ciphers.

The paper by Edwar Gomez et al. [25] presents a comprehensive study on the hardware implementation of the PRESENT encryption algorithm, detailing its design, implementation, and testing. The findings indicate a 30% reduction in hardware usage compared to earlier FPGA implementations. Additionally, the use of standard VHDL programming resulted in a 26.89% decrease in resource usage on a Xilinx Spartan 3AN FPGA.

Kiran Kumar.V.G et al in [31], implemented TEA,XTEA,PRESENT and IDEA on Verilog Xilinx 14.2 the target device is xc2v40-5cs144 and compared these 80-bit key PRESENT with other three 128-bit key lightweight algorithms with respect to the parameters Maximum Frequency(MHz), Time delay(ns), LUT, From this implementation the authors concluded that PRESENT uses less area, takes lesser time to encrypt a 64-bit data block and PRESENT is a better cipher in terms of area, security and performance[31].

Panasayya Yalla and Jens-Peter Kaps in [32] implemented lightweight cryptography for FPGAs by introducing block cipher independent optimization techniques for Xilinx Spartan3 FPGAs and applying them to the lightweight cryptographic algorithms HIGHT and PRESENT. Implementation of PRESENT consumed less than 120 slices, encrypts data at 28.46 Mbps and has a throughput over area ratio 0.24, Maximum Delay of 8.78ns. The authors compared the results for PRESENT with that of the other ciphers like Camelia, TinyXTEA-3, HIGHT and AES which were implemented by the same researchers in their prior projects.

Edwar Gomez and colleagues in [33] executed the PRESENT algorithm on Xilinx FPGA, an embedded hardware platform, utilizing the ISE development tool and cost-effective FPGAs like Spartan 3AN and Spartan 3E. Their study emphasized hardware platform efficiency, resulting in four implementations across four different microcontrollers. The PRESENT algorithm's performance

on various FPGA-type programmable logic devices was evaluated using encryption and decryption functions with an 80-bit maximum password size. Efficiency was assessed using 64-bit block vectors, with throughput and occupied slices (CLB-GE) serving as the chosen evaluation metrics. This implementation achieved a 26% decrease in hardware usage compared to previous versions. The maximum delay recorded was 6.865 ns, corresponding to a peak operational frequency of 145.6 MHz.

Factors	**PRESENT**
Created By	Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe
Expansion	---
Key Size(bit)	80,128
Block Size(bit)	64
Rounds	31
Cipher Type	Block Cipher
Structure Type	SPN

Table 4.3: Characteristics of PRESENT

Sl. No	Paper	Method of Implementation	Block Length	Key Length	Year
1	[16]Present: An Ultra-Lightweight Block Cipher	Simulation Mentor Graphics Modelsim SE PLUS 5.8c	64bits	80bits	2007
2	[25] Implementation Of The Cryptographic Algorithm “Present” In Different Microcontroller Type Embedded Software Platforms	ARM platform in Cortex-M0 family	64bits	80bits	2017
3	A Performance Comparison Of The Present Lightweight Cryptography Algorithm On Different Hardware Platforms	A standard laptop, a BeagleBone Black and a Xilinx Zynq-7000	64bits	80bits	2020
4	An Efficient Implementation Of Present Cipher Model With 80 Bit Nd 128 Bit Key Over Fpga Based Hardware Architecture	FGCA	64,128bits	80bits	2018
5	Threshold Implementations Of The Present Cipher	FGCA Verilog And synthesized for Virtex-5	64bits	80bits	2017

Table 4.4: Literature Review of various implementations of PRESENT cipher

4.2.5 Communication Protocols

The communication protocol used in the implementation of PRESENT is the same as that of IDEA and the details of the communication protocols are explained in the previous chapter.

4.2.6 Methodology

IMPLEMENTATION Using Cooja Simulator

Contiki is a popular, free and open-source operating system for IoT programming available under the BSD licence with the base code of the C programming language. Contiki can be used for communication between low powered RFID chips in wireless networks with a high degree of performance and security. IoT based wireless networks such as Smart Homes, Smart Cities and Smart Grids etc use devices with constrained resources like sensors, RFID Tags, routers and gateways. Several tiny Operating Systems are available for IoT and under the open-source category, Contiki is the renowned OS with the base of C Language. The performance analysis of IoT based smart home networks can be studied efficiently by simulating the networks in the Cooja Network Simulator of Contiki. In this project, we studied the performance analysis of the smart home networks with 15 nodes and 30 nodes with and without presence of PRESENT cryptography algorithm. Contiki has built-in codes in C and works on both IPv4 and IPv6. The integration of lightweight protocols like CoAP, MQTT, RPL and XMPP etc has enhanced the security aspects of Cooja Simulator. But in an IoT network, the data path between the sensor nodes and gateways are still un-encrypted and pose for several security threats.

The programming on Contiki is done using the Cooja network simulator, in which the base libraries of RFID chips and sensors are available in C. To program, control and monitor the remote IoT devices, the back-end C programs and related header files can be customised and recompiled to get the desired results. Contiki works on IPv4 as well as IPv6 networking with the integration of lightweight protocols, so that low power chips and radio frequency chips can be connected without performance issues.

Simulation setup: The various security algorithms to be deployed in SHNs will be simulated using the Cooja network simulator. Instant Contiki, which includes the Cooja network simulator, can be installed on a virtual Linux machine. Cooja also contains the Contiki source code. It is a Java-based simulator utilized in this work. Contiki is open-source, and the code is available for editing. The discrete event Cooja simulator has numerous built-in radio mediums, such as UDGM (Unit Disk Graph Medium) and MRM (Multi-path Ray-tracer Medium). It was developed at SICS to assist in debugging and testing Contiki OS-based systems. The proposed system will be implemented in Contiki and compiled for the MSP430 processor microcontroller. We used the Sky mote sensor devices for our simulations.

A simulated node in Cooja has three basic properties: its data memory, the node type, and its hardware peripherals. The node type can be shared among several nodes and determines properties common to all these nodes. For example, nodes of the same type run the same program code on the same simulated hardware peripherals and are initialized with the same data memory.

4.2.7 Performance Analysis: The performance metrics to analyse the performance were taken same as the performance metrics used for implementation of IDEA which is described in the previous chapter.

Performance	With PRESENT		Without PRESENT	
	15Nodes	30Nodes	15Nodes	30Nodes
Throughput ((bps)	38.25	60.8	45.35	52.5
PDR	95.57	91.93	97.56	94.45
Delay(ms)	53.7	22.18	22.5	23.6
Energy consumption(joules)	0.00451	0.00653	0.00150	0.00167
Execution Time(s)	5.48	6.7	3.37	4.45

Table 4.5: performance analysis of Smart home network with and without PRESENT.

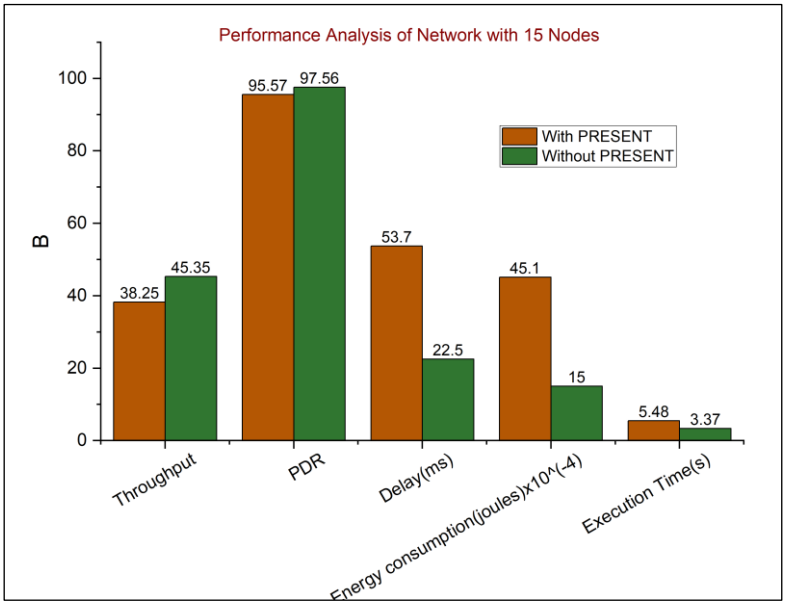


Figure 4.6: Performance graph of smart home network with 15 Nodes

Throughput: Here "Throughput" measures the amount of data successfully transmitted per unit of time (bps). The simulation results with 15 nodes are illustrated in the graph Figure 12. This graph indicates that the implementation of PRESENT yields only a negligible increase in throughput, suggesting that using PRESENT does not compromise the efficiency of data transmission within the network.

Packet Delivery Ratio (PDR): The implementation of PRESENT shows that the PDR remains nearly unchanged compared to the PDR without any cryptographic algorithm. This finding indicates that PRESENT does not cause unnecessary packet loss, resulting in more reliable data delivery.

Delay: There is a slight increase in delay with the implementation of PRESENT, likely due to the additional processing time required for encryption

and decryption operations. However, this increase in delay is relatively minor when compared to the negligible changes in throughput and PDR.

Energy Consumption: The use of PRESENT results in a slight increase in energy consumption, as encryption and decryption operations require energy. However, this increase is minimal and may be overshadowed by the benefits of improved security and reliability.

Execution Time: A minor increase in execution time is observed with PRESENT, attributable to the computational overhead of encryption and decryption. However, this slight increase does not significantly affect the network's overall performance.

Overall, the graph suggests that implementing the lightweight cryptography algorithm PRESENT in the IoT-based smart home network has a positive impact on overall performance. It enhances security by ensuring data confidentiality and integrity while maintaining acceptable levels of throughput, PDR, delay, energy consumption, and execution time.

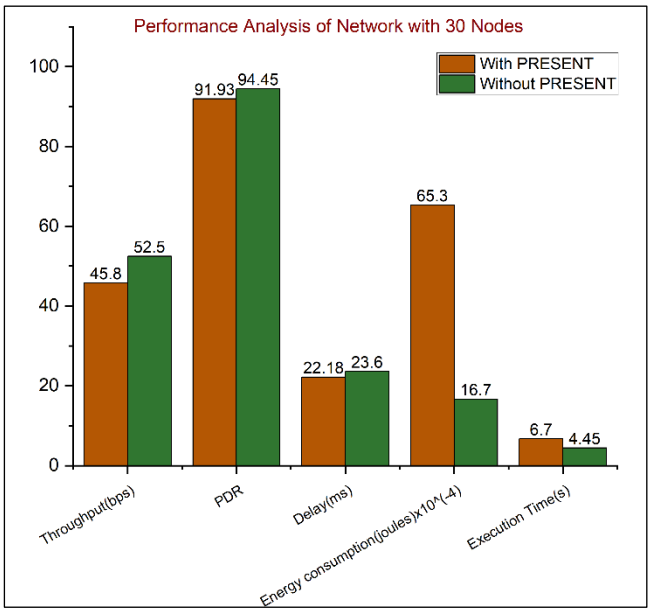


Figure 4.7: Performance graph of smart home network with 30 Nodes

Based on this graph, the implementation of the PRESENT LWC algorithm in this smart home network with 30 sensors results in:

- Reduced Throughput with PRESENT. This is predictable as processing of encryption and decryption adds extra overhead, consuming processing time and bandwidth, thus reducing the raw throughput.
- PDR is slightly *lower* with PRESENT but insignificant. But during simulation its observed that the packet loss is very minimal.
- Delay is slightly lower with PRESENT. Adding security features to the network has a slight impact on network congestion.
- Energy consumption is intensely *higher* with PRESENT. This is a critical factor for battery-powered IoT devices. Cryptographic operations are

computationally intensive, leading to increased energy usage. This is a significant trade-off for the added security.

4.2.8 Conclusion

The implementation of Light Weight Cryptography algorithm PRESENT using Cooja Simulator in Instant Contiki OS, has demonstrated that the adding of cryptographic security to an IoT network do not trade-off the performance of the network to higher extent. While PRESENT offers security benefits, the trade-offs in terms of throughput, energy consumption, and execution time need to be carefully considered based on the specific requirements of the smart home application. Further research and optimization might be necessary to make PRESENT more suitable for resource-constrained IoT deployments.

4.3 IMPLEMENTATION OF LWC Algorithm HIGHT

4.3.1 Introduction

Deukjo Hong et al., scientists from CIST-Korea University, the University of Seoul, the National Security Research Institute (NSRI), and the Korea Information Security Agency (KISA), developed a block cipher called HIGHT (HIgh Security and light weiGHT) with a 64-bit block length and a 128-bit key length. This algorithm was implemented using only 3,048 gates on 0.25μm technology and consists of just 32 iterations involving simple operations such as XOR, addition modulo 28, and left-wise bit rotation. As a result, it is suitable for low-cost, low-power, and ultra-light implementations[34].

The major feature of HIGHT is that it doesn't include any S-box in the encryption process. The plaint text is divided into 8 subtexts of 8-bit length and the 128-bit master key is divided into 16 sub-keys of 8 bits each. The constant generation algorithm generates 8 whitening keys and 128 sub-keys from the master keys. The first 4 whitening keys are used for initial transformation and the last four keys are used for final transformation of the plain text . A 7-bit Linear Feedback Shift Register(LFSR) is used for constant generation algorithm.

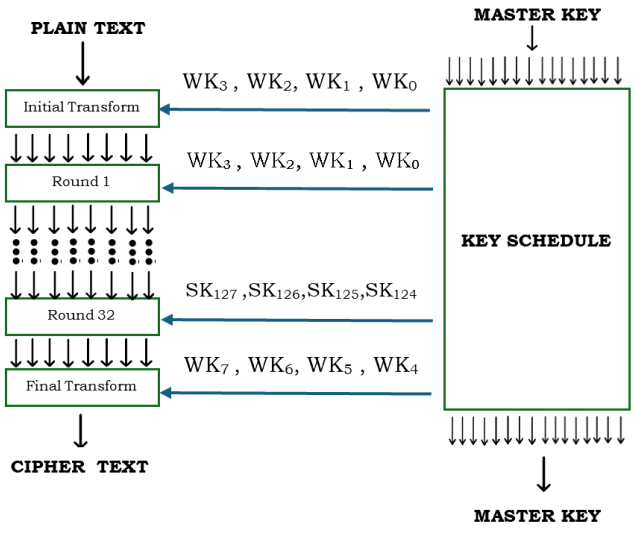


Figure 4.8: Encryption process of HIGHT

4.3.2 Review of Literature

The HIGHT algorithm was proposed, designed and developed by Deukjo Hong et al in their paper [34]. In this article, the authors compared the hardware implementation of HIGHT with that of AES and proved that HIGHT needs a smaller number of gates and thus suitable for resource constrained ubiquitous devices.

Jeremy Aguilar et al in their paper [35], studied the time and energy requirements for HIGHT encryption algorithm implementation on a 8-bit micro controller XILINX Spartan-3 and for synthesization VHDL was used.

Bohun Kim et al. in the article[36] studied various features of HIGHT, such as code size, memory usage, and execution time in optimized implementations on IoT platforms. They utilized 8-bit AVR and 32-bit ARM Cortex-M3 for hardware implementations.

The researchers employed a Look-Up Table (LUT) to replace bitwise operations in the delta update, F0, and F1 functions. The LUT is stored in 8-bit alignment, and the registers are efficiently allocated, thereby reducing the frequency of memory access. Additionally, they introduced a master key management scheme in the key schedule, which rotates the master keys and selects appropriate keys for each round.

Bassam Jamil Mohd et al. [37] suggested in their paper that pipeline designs should be replaced with scalar designs because scalar designs take up less space and consume less energy. They stated that their experiments showed that a scalar design with one and eight rounds can achieve the best performance compared to a pipeline design with two and four rounds. They also stated that 18% resource and 10% energy savings can be achieved with the scalar design.

4.3.3 Communication Protocols

The communication protocol used in the implementation of HIGHT is the same as that of IDEA and the details of the communication protocols are explained in the chapter 4.1.

4.3.4 Simulation

The communication Protocol used is Message Queuing Telemetry Transport (MQTT) and all other experiments and simulations are performed exactly like that of IDEA algorithm.

4.3.5 Performance Analysis

Performance	**WITH HIGHT**	**WITHOUT HIGHT**	**WITH HIGHT**	**WITHOUT HIGHT**
	15 Nodes	15 Nodes	30 Nodes	30 Nodes
Throughput(bps)	81.2	68.6	151.4	124.8
Packet Delivery Ratio	83.54	94.56	75.68	90.82
Delay(ms)	48.86	34.54	52.56	45.35
Energy consumption(joules)x10^-5	72.9	65.76	98.55	85.88
Execution Time(s)	4.37	3.55	8.26	6.76

Table 4.6: Performance analysis of network with and without HIGHT

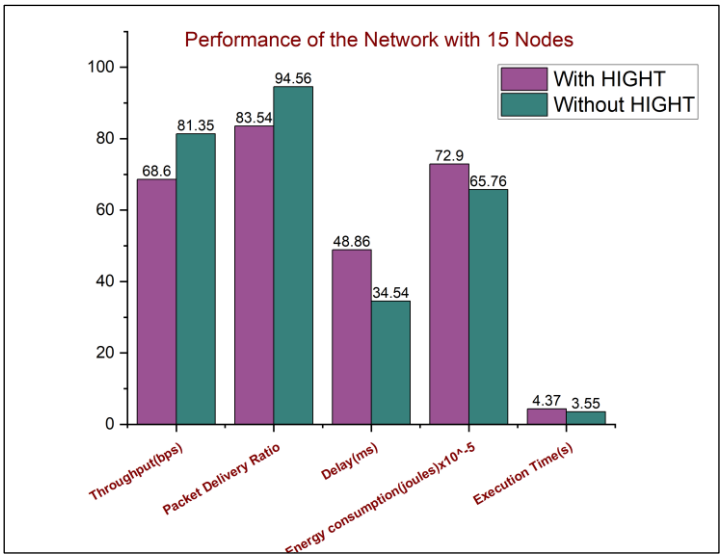


Figure 4.9: Performance analysis of network with and without HIGHT

The data in the Table 4.6 and the graph in the Figureure 4.9 compare the performance of a 15-nodes IoT-based smart home network with and without the HIGHT lightweight cryptographic algorithm implemented in Cooja simulator to encrypt messages from sensors to the root node or gateway. The various metrics listed here interpret that implementation of light weight security mechanisms do not attenuate the performance of the network. It is observed that Throughput is lower with HIGHT. It is obvious because the cryptographic operations add overhead, consuming processing time and bandwidth, thus reducing the throughput. PDR is significantly lower with HIGHT. This indicates that the larger packet size—caused by the additional cryptographic headers could lead to a noticeable increase in packet loss, thereby affecting reliability. We suggest that this issue can be traded off because the security concerns are of higher importance than the speed of the communication. Similarly, the increase in delay, latency, energy consumption and execution time are all anticipated as cryptographic operations add processing time.

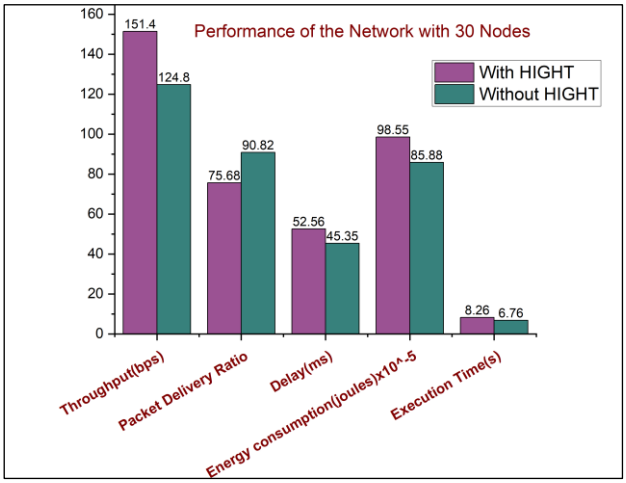


Figure 4.10: Performance of the Network with and without HIGHT

The graph in the Figure 4.10 compares the performance of a 30-nodes IoT-based smart home network with and without the HIGHT lightweight cryptographic algorithm. All the performance metrics show similar behaviours as that of network with 15-nodes with obviously augmented values for a more complex network topology.

4.4 Detection and Mitigation of DDoS Attack in SHNs

4.4.1 Introduction

Regardless of the many benefits that can be gained from IoT-based smart home networks, these networks are highly vulnerable to attacks. One, among thousands of malware types, the Distributed Denial of Service (DDoS) attack, can cause significant damage and disruption to a network. If a DDoS attack compromises legitimate nodes, it can flood the network with messages, causing the entire system to come to a halt, including the router and all other connected nodes. One of the DDoS attacks that an attacker usually uses to damage a network service is a “Hello” message flood attack. A Hello message flood attack is a type of Distributed Denial of Service (DoS) attack that targets IoT networks. In this attack, an attacker compromises any random node in the network. And then the compromised node uninterruptedly sends "Hello" messages to other nodes. The disproportionate volume of these messages consumes significant network bandwidth and processing power of the legitimate nodes. As a result, legitimate nodes struggle to process the flood of messages, leading to degraded network performance or even complete network failure. At the network layer, the IoT network is safeguarded by the 6LowPAN and RPL protocols. In this project we have implemented Intrusion Detection System with RPL. This project implements an Intrusion Detection System AtRouterIDS(ARIDS) to detect DDoS attacks in IoT-based smart home networks. The ARIDS detects and prevents at Network layer with RPL protocol.

4.4.2 Overview of RPL protocol

Smart home networks are based on IoT that consist of resource constrained tiny devices like sensors an actuator (Nodes/Motes). RPL is a distance vector protocol which uses IPv6 addressing policy and these networks do not end up in a loop when the new nodes connect to the network. In RPL based networks, the new nodes form a Destination Oriented Directed Acyclic Graph (DODAG) that routes the packets to a single node called DODAG root node or server node.

4.4.3 DDoS attack in IoT-based smart home networks

A distributed denial of service (DDoS) attack aims to partly or totally disable a targeted server by flooding it with internet traffic. The primary goal of this attack is to interrupt the normal flow of traffic to the victim's server or network. DDoS attacks are classified as volumetric attacks and frequently compromise low-security IoT devices, such as webcams, baby monitors, and smoke detectors. These compromised devices generate high traffic that is redirected to servers, disrupting their normal operations. IoT devices, including CCTV cameras, webcams, baby monitors, and smartwatches,

typically have poor built-in security and are constrained by low computational power, limited storage, and restricted battery capacity. Due to these vulnerabilities, attackers can use malware tools like the BashLite, Reaper, Mirai BOT or Lizard-Stresser to gain control of these devices[38].
Types of DDoS attacks

4.4.4 Review of Literature

In the article [39], Tayyab Mahamood has demonstrated the working of routing and compression protocol in real time simulation using Contiki Based COOJA Simulator. This author has demonstrated the implementation of RPL in the simulator and explained the evaluation process of various performance metrics. This article has helped our project to finalise the evaluation metrics required to compare the performance of the smart home network with and without IDS.
[40] is a project which focussed on anomaly based IDSs and the authors demonstrated the major detection and mitigation techniques using machine learning algorithms.
S. Cakir et al in their work [41] have developed GRU based Deep learning model to detect and mitigate attacks on RPL based IoT networks.

4.4.5 Evaluation Metrics

To evaluate the performance of the smart home network with ContikiRPL protocol, several protocol parameters are available namely Convergence Time, Average Power Consumption, Latency, PDR(Packet Delivery Ratio), Radio Duty Cycle interval, Network Hops, number of DIS(DODAG Information Solicitor) and DIO(DODAG Information Object) formed etc. These parameters are called Performance Metrics. Along these, some other metrics also significantly affect the performance of network. And they are the number of nodes, power consumption by each node, network topology, the number of sink nodes and the type of mobility etc. Network scalability also increases all metrics but results in less packet loss. In this experiment we have considered the Objective Function ETX(Expected Transmission Count) and studied power consumption, PDR and Latency.
Packet Delivery Ratio: PDR is defined as the number of received packets at the sink node to the number of packets sent to the sink node. The average PDR is the number of successfully received packets at the sink node divided by the number of packets sent from all the nodes to the sink node .

$$\textbf{\textit{Average PDR = (Total number of packets received)/(Total Packets sent)*100}}$$

Latency: The difference between the packet sent time and packet received time is called the Latency. In other words, Network Latency (End-to-End delay) is considered as the total time taken for a packet to travel from client node to the root node (destination).

$$\textbf{\textit{Total Latency}} = \sum \textbf{\textit{Rvcd Time}} - \textbf{\textit{Sent Time}}$$

$$\textbf{\textit{Average Latency = Total Latency/Total Packets received.}}$$

To compute the power consumption, we have used the conventional PowerTracker app available in Contiki. This app estimates the energy used for CPU processing, packet transmission and listening.

Power Consumption:
The amount of the energy consumed by each node in the network is regarded as the power consumption. It is the sum of energy consumed by each node in the network. There are four factors of power consumption being recorded in Cooja simulator namely Low Power Mode or LPM(during sleep mode of the sensors/nodes), CPU(during processing) , radio-listen (power consumed to receive data) and radio-transmit (power consumed to transmit data). The total power consumed by each node is the sum of all these four factors. In RPL based networks, the node which is closer to the DODAG root consumes more power because it must receive, process and transmit the packets of its sub-nodes also. The power consumed by each node is calculate as follows[42]:

$$\boldsymbol{Energy(mJ} = \Big((\boldsymbol{CPU_{Power}} * \boldsymbol{CPU_{Time}}) + (\boldsymbol{LPM_{Power}} * \boldsymbol{LPM_{Time}}) + \boldsymbol{T_{x\_time}} * \mathbf{17.4} + \boldsymbol{R_{x\_time}} * \mathbf{18.8}\Big) * \mathbf{3}\boldsymbol{v}$$

Where:

- **CPU_power:** Power consumption of the CPU in milliwatts (mW)
- **CPU_time:** Time spent in active CPU mode (seconds)
- **LPM_power:** Power consumption in Low Power Mode (LPM) in milliwatts (mW)
- **LPM_time:** Time spent in LPM (seconds)
- **Tx_power:** Power consumption during transmission in milliwatts (mW) which is 17.4mW
- **Tx_time:** Total transmission time (seconds)
- **Rx_power:** Power consumption during reception in milliwatts (mW) which is 18.8mW
- **Rx_time:** Total reception time (seconds)
- **v:** Voltage (in volts)

4.4.6 Methodology:

The project is implemented in the Cooja Simulator, featuring a smart home network with four different frameworks. Each framework includes two test cases. In the first framework, we have 15 normal nodes and one sink node. The sink node, or root node, serves as the border router, communicating with the Internet. The other nodes, which are the sensors connected to household appliances, do not communicate directly with the Internet. Rather, they interact with the border router, which then interfaces with the Internet.
The other three Frameworks are designed by varying the number of normal nodes and malicious nodes, with only one sink node present. The two test cases are designed to compare the network's performance of the network with respect to the radio range of the sink node. In test case A, the malicious nodes are kept outside the radio range of the sink node, while in test case B, the malicious nodes are kept within the radio range of the sink node. The table below provides a consolidated topology of the four frameworks with two test cases each.

Framework	Test Case	Nodes/Motes	Radio Range
1	A	14 Normal Nodes;1 Sink Node; No malicious Nodes	50% of normal nodes are **outside** the radio frequency range of the sink node
	B	14 Normal Nodes;1 Sink Node; No malicious Nodes	All normal nodes are **inside** the radio frequency range of the sink node
2	A	13 Normal Nodes;1 Sink Node; 1 malicious Nodes	Malicious node is outside the radio frequency range of the sink node
	B	13 Normal Nodes;1 Sink Node; No malicious Nodes	Malicious node is inside the radio frequency range of the sink node
3	A	11 Legitimate nodes; 1 Sink Node; 3 Malicious nodes	All 3 Malicious nodes are outside the radio frequency range of the sink node
	B	11 Legitimate nodes; 1 Sink Node; 3 Malicious nodes	All 3 Malicious nodes are within the radio range of the sink node
4	A	12 Legitimate nodes; 1 Sink Node; 1 Malicious Nodes; 1 IDS node	The IDS node is outside the radio range of the Sink Node
	B	12 Legitimate nodes; 1 Sink Node; 1 Malicious Nodes; 1 IDS node	The IDS node is inside the radio range of the Sink Node

Table 4.7: List of topologies for simulation

The simulation configuration for these Frameworks is as follows:

Specification	Value
Operating System	Instant Contiki 3.0 Installed on VMWare Workstation 17 Player
Simulation Area/Range of nodes	10x10Sqm
Radio Medium Model	UDGM(Unit Disk Graph Medium)-Distance Loss
Network Protocol	Contiki RPL
Simulation time	600seconds
Mote	Skymote
Packet Size	512Bytes
Topology	Random Positioning

Table 4.8: Simulation Configuration

4.4.7 Results and Analysis:

4.4.7.1 Framework 1 Test Case A:

In this simulation set up, all the types of nodes are randomly deployed in the simulation area and their location doesn't change during runtime. The root node is responsible for initiating the network and gathering the data. The root node does not suffer power issues whereas the sub-nodes are powered by batteries .

The Network Topology is star-like topology and here Sky mote 1 functions as the gateway node. RPL often promotes energy-efficient routing by forming trees or directed acyclic graphs (DAGs) where nodes forward traffic towards a designated root, the node 1. The smart home network topology is shown below:

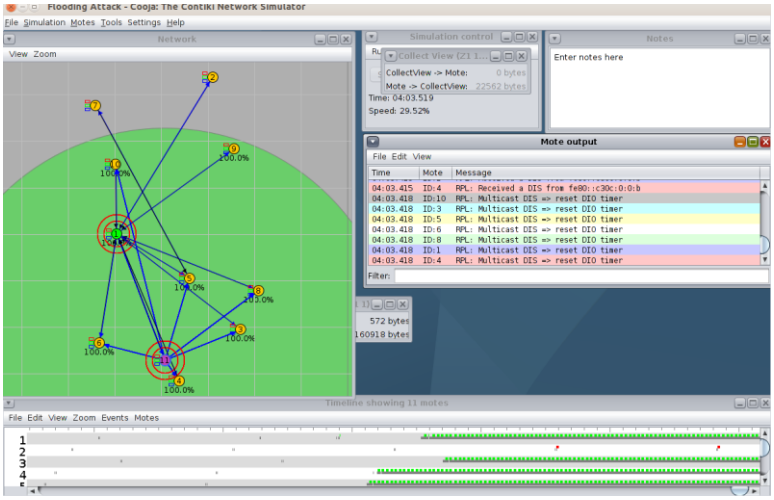
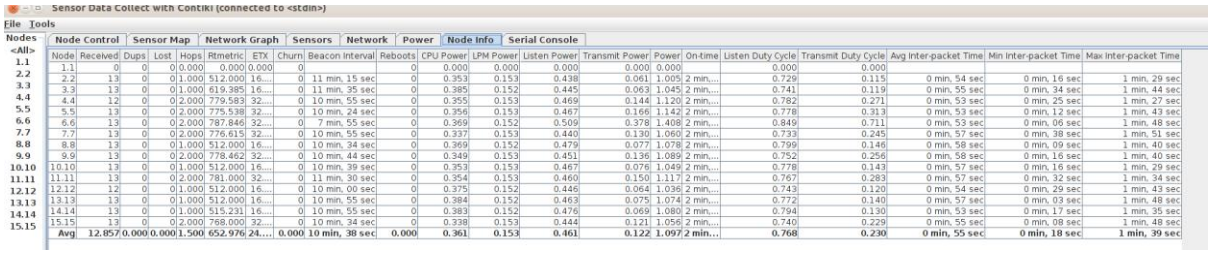


Figure 4.11: Framework1 Test Case A- 10 Normal Nodes;1 Sink Node; No attacker node

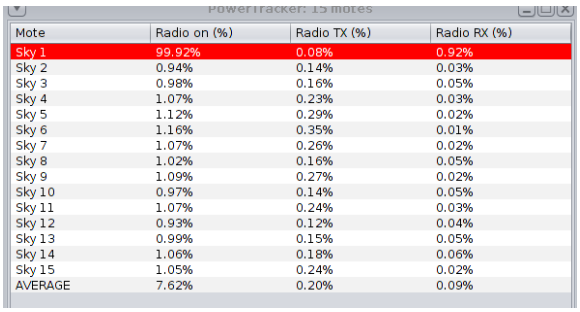
Here we have kept 7 nodes out of 14 legitimate nodes outside the radio range of the sink node and other 7 nodes inside the radio range. The node info taken during the simulation gives the parameters like data packets received, data packets lost, number of hops or Hop-Count, ETX, CPU Power, LPM Power Average Power consumed, Tx, Rx, Radio duty cycles etc.



Node	Received	Dups	Lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power	Transmit Power	Power	On-time	Listen Duty Cycle	Transmit Duty Cycle	Avg Inter-packet Time	Min Inter-packet Time	Max Inter-packet Time
1.1	0	0	0	0.000	0.000	0.000	0		0	0.000	0.000	0.000	0.000	0.000		0.000	0.000			
2.2	13	0	0	1.000	512.000	16....	0	11 min, 15 sec	0	0.353	0.153	0.438	0.061	1.005	2 min,...	0.729	0.115	0 min, 54 sec	0 min, 16 sec	1 min, 29 sec
3.3	13	0	0	1.000	619.385	16....	0	11 min, 35 sec	0	0.385	0.152	0.445	0.063	1.045	2 min,...	0.741	0.119	0 min, 55 sec	0 min, 34 sec	1 min, 44 sec
4.4	12	0	0	2.000	779.583	32....	0	10 min, 55 sec	0	0.355	0.153	0.469	0.144	1.120	2 min,...	0.782	0.271	0 min, 53 sec	0 min, 25 sec	1 min, 27 sec
5.5	13	0	0	2.000	775.538	32....	0	10 min, 24 sec	0	0.356	0.153	0.467	0.166	1.142	2 min,...	0.778	0.313	0 min, 53 sec	0 min, 12 sec	1 min, 43 sec
6.6	13	0	0	2.000	787.846	32....	0	7 min, 55 sec	0	0.369	0.152	0.509	0.378	1.408	2 min,...	0.849	0.711	0 min, 53 sec	0 min, 06 sec	1 min, 48 sec
7.7	13	0	0	2.000	776.615	32....	0	10 min, 55 sec	0	0.337	0.153	0.440	0.130	1.060	2 min,...	0.733	0.245	0 min, 57 sec	0 min, 38 sec	1 min, 51 sec
8.8	13	0	0	1.000	512.000	16....	0	10 min, 34 sec	0	0.369	0.152	0.479	0.077	1.078	2 min,...	0.799	0.146	0 min, 58 sec	0 min, 09 sec	1 min, 40 sec
9.9	13	0	0	2.000	778.462	32....	0	10 min, 44 sec	0	0.349	0.153	0.451	0.136	1.089	2 min,...	0.752	0.256	0 min, 58 sec	0 min, 16 sec	1 min, 40 sec
10.10	13	0	0	1.000	512.000	16....	0	10 min, 39 sec	0	0.353	0.153	0.467	0.076	1.049	2 min,...	0.778	0.143	0 min, 57 sec	0 min, 16 sec	1 min, 29 sec
11.11	13	0	0	2.000	781.000	32....	0	11 min, 30 sec	0	0.354	0.153	0.460	0.150	1.117	2 min,...	0.767	0.283	0 min, 57 sec	0 min, 32 sec	1 min, 34 sec
12.12	12	0	0	1.000	512.000	16....	0	10 min, 00 sec	0	0.375	0.152	0.446	0.064	1.036	2 min,...	0.743	0.120	0 min, 54 sec	0 min, 29 sec	1 min, 43 sec
13.13	13	0	0	1.000	512.000	16....	0	10 min, 55 sec	0	0.384	0.152	0.463	0.075	1.074	2 min,...	0.772	0.140	0 min, 57 sec	0 min, 03 sec	1 min, 48 sec
14.14	13	0	0	1.000	515.231	16....	0	10 min, 55 sec	0	0.383	0.152	0.476	0.069	1.080	2 min,...	0.794	0.130	0 min, 53 sec	0 min, 17 sec	1 min, 35 sec
15.15	13	0	0	2.000	768.000	32....	0	10 min, 34 sec	0	0.338	0.153	0.444	0.121	1.056	2 min,...	0.740	0.229	0 min, 55 sec	0 min, 08 sec	1 min, 48 sec
Avg	12.857	0.000	0.000	1.500	652.976	24....	0.000	10 min, 38 sec	0.000	0.361	0.153	0.461	0.122	1.097	2 min,...	0.768	0.230	0 min, 55 sec	0 min, 18 sec	1 min, 39 sec

Figure 4.12: Node info of Frame Work1-Test Case A

We have used the built-in App PowerTracker to analyse the activities of the nodes and the Gateway. The output obtain for this framework is shown in the Figure.



Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	99.92%	0.08%	0.92%
Sky 2	0.94%	0.14%	0.03%
Sky 3	0.98%	0.16%	0.05%
Sky 4	1.07%	0.23%	0.03%
Sky 5	1.12%	0.29%	0.02%
Sky 6	1.16%	0.35%	0.01%
Sky 7	1.07%	0.26%	0.02%
Sky 8	1.02%	0.16%	0.05%
Sky 9	1.09%	0.27%	0.02%
Sky 10	0.97%	0.14%	0.05%
Sky 11	1.07%	0.24%	0.03%
Sky 12	0.93%	0.12%	0.04%
Sky 13	0.99%	0.15%	0.05%
Sky 14	1.06%	0.18%	0.06%
Sky 15	1.05%	0.24%	0.02%
AVERAGE	7.62%	0.20%	0.09%

Figure 4.13: PowerTracker output of the Framework 1 -Test Case A

Here Radio On (%) shows the percentage of the time during which the radio of the sensor was actively turned on. The mote Sky1 is the sink node or router node. All other nodes communicate to the internet through this mote. Hence this node's radio is always active and that's why it shows 99.9% Radio on time

for Sky1. A higher percentage generally indicates more active communication and potentially higher energy consumption.
The parameter Radio TX (%) represents the percentage of time the mote's radio was actively transmitting data. This is crucial for understanding the communication load and energy consumption associated with sending data. Radio RX (%) represents the percentage of time the mote's radio was actively receiving data. This metric is important for assessing the amount of incoming traffic and the energy consumption associated with receiving data.

Sensor Map

In the Cooja Simulator the **sensor map** represents a visual depiction of the physical layout of the IoT network, usually with positions and the connections of the nodes in the network. It helps to understand the interactions between different devices.
In this Framework we have **1 sink node** and **14 UDP sender nodes**, the sensor map represents the following layout.

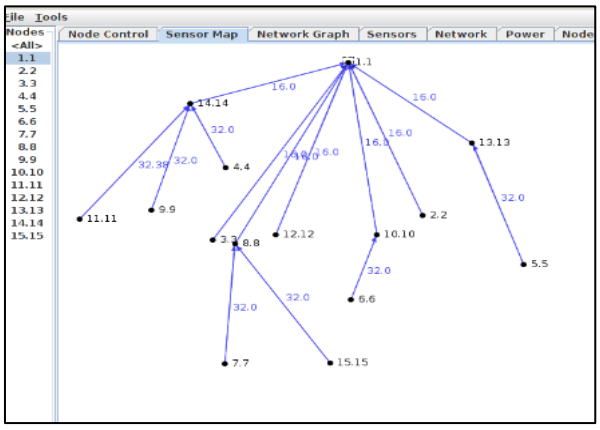


Figure 4.14: Sensor Map for Frame Work1

1. **Sink Node (Root of RPL DAG)**:
    - The sink node is usually the central node and is connected to the internet and acts as the data collection point. It is the root of the RPL Directed Acyclic Graph (DAG).
    - The sink node is often displayed at a fixed position and connected to other nodes based on RPL's routing protocol.
2. **UDP Sender Nodes (Leaf or Intermediate Nodes)**:
    - These nodes generate and send UDP packets to the sink node.
    - In the sensor map, these nodes may be scattered in a grid or random pattern, depending on the simulation setup, and they communicate with the sink node.
3. **Connectivity**:
    - The sensor map also shows the communication range of each node (usually depicted with circles around each node). A node can communicate directly with neighbouring nodes within its transmission range. If direct communication isn't possible, RPL

routes the data through intermediate nodes (multi-hop communication).

- The map visually illustrates these relationships, helping to understand which nodes are communicating directly and which are relaying data to other nodes.

Network Graph

The network graph represents the logical and hierarchical structure of IoT network. In this Framework, we get the visual depiction of the communication between the sink or router node(root of the RPL DAG) and other 14 UDP-sender nodes. The graph highlights the hierarchical structure established by the RPL with potential insights into network connectivity, hop count and the possible congestion points based on the edge thickness and node positions within the graph.

Node Connectivity

- **Edges (Links)**: Lines connecting nodes represent the established communication links between them, indicating the routing path data takes to reach the sink node. This shows direct or indirect communication paths.

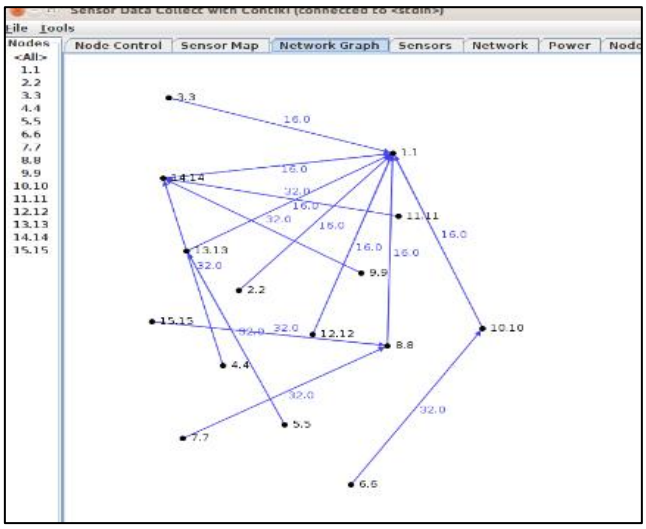


Figure 4.15: Network Graph for Framework1 – Test Case A

- **Edge thickness:** The thickness of an edge visually represents the amount of data traffic flowing through that link, with thicker lines signifying higher traffic volume.
- **Connectivity Patterns**: The graph should show full connectivity between all sender nodes and the sink, indicating successful data transmission and proper network formation. Observing how nodes are connected helps understand the network's resilience and efficiency.
- **Hop Count Distribution:** By examining the number of hops between sender nodes and the sink, the graph reveals the potential network congestion areas where a high number of nodes are routing through the same few hops.

RPL DAG (Directed Acyclic Graph)

- The graph shows the hierarchical structure where each node's rank determines its position, demonstrating the inherent characteristic of RPL. The sink node is at the top, and other nodes are positioned based on their rank.
- **Preferred Parent**: Each node selects a preferred parent, which determines the primary route for data transmission to the sink node. This is crucial for understanding the routing paths and data flow.
- In this framework we observed that the DODAG formation is undisturbed as there is no malware nodes present and the network communication is not under attack.

Average Power Consumption

The Power Consumption Graph generated during network simulation shows the power consumption breakdown into four components: LPM, CPU, Radio Listen, and Radio Transmit. LPM (Low Power Mode) represents the power consumed when the node is in a low-power state. CPU refers to the power consumed by the node's processor. Radio Listen indicates the power consumed while the node is listening for incoming signals, and Radio Transmit represents the power consumed during data transmission. The power consumption of each UDP sender node spikes during transmission when it sends data to the sink, as transmission consumes the most energy in the entire communication cycle. When sender nodes are not transmitting or listening for incoming control messages such as DIO or DAO, power consumption is moderate.

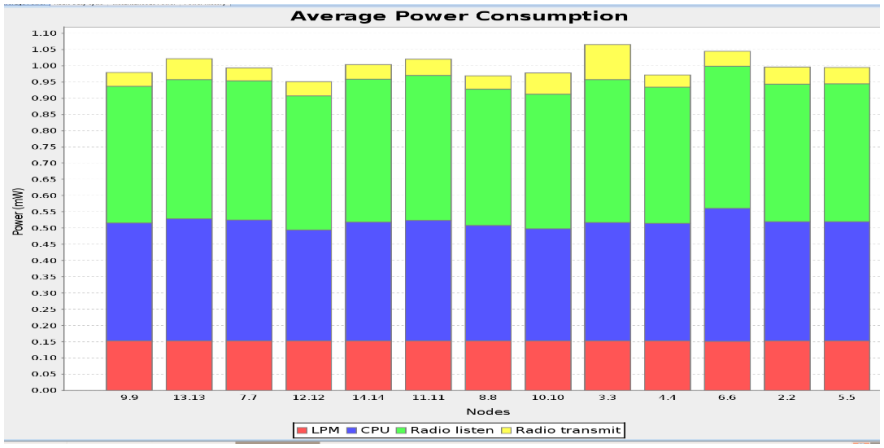


Figure 4.16: Average Power Consumption in Framework1 with Test Case A

Power consumption varies significantly between different nodes. Some nodes exhibit higher power consumption due to increased radio transmission, while others consume less power due to reduced activity. Nodes farther from the sink or with weaker links may require higher transmission power, leading to increased energy consumption. Nodes with a higher traffic load (e.g., those closer to the sink or with more neighbours or nodes attacked by malware) will have greater radio transmit and listen power consumption. Additionally, nodes actively participating in RPL operations (e.g., DIO transmission, parent selection) will experience higher CPU and radio power consumption. Frequent

wake-up events and transitions between sleep and active modes can also consume significant energy. Nodes with lower power consumption are likely those with lower traffic loads, shorter distances to the sink, and effective sleep/wake-up mechanisms. In this framework, there are no malicious nodes; therefore, no node shows extreme variations in power consumption compared to others.

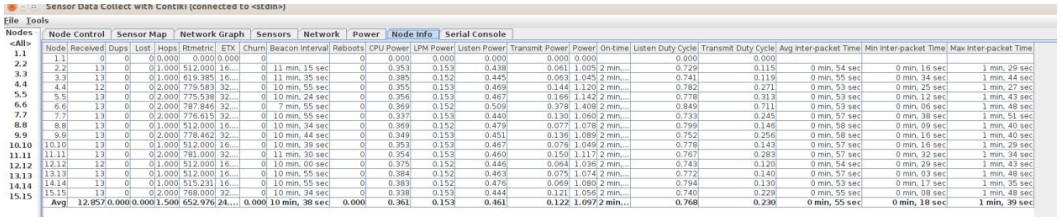


Figure 4.17: Node Info of Framework1 with Test Case A

The performance of the framework is studied, and the following data has been recorded.

Parameter	Value
PDR	99.3%
Latency(s)	3.5
Average Power Consumed	1.097mV

Table 4.9: Performance of the network with Framework1 -Test Case A

4.4.7.2 Framework 1: Test case B

Here we have kept all 14 legitimate nodes inside the radio frequency range of the route node and the performance of the network is studied exactly alike the Test Case A. The performance of the framework1 with Test Case B is studied, and the following data has been recorded. Here the Average power consumed by all the nodes is lower compared to the Test Case A because all of the nodes are using only one hop count and the distance between the UDP-sender nodes and the sink node is less as compared to that of the Test Case A. The node Info collected during the simulation process is shown in the figure and the performance metric is tabulated in the table:

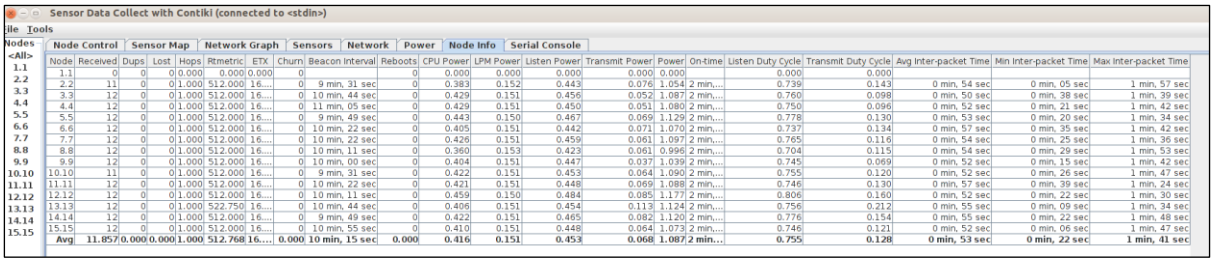


Figure 4.18: Node Info of Framework1 with Test Case B

Parameter	Value
PDR	99.8%
Latency	3.2
Average Power Consumed	1.087mV

Table 4.10: Performance metrics of Framework1 with Test Case B

4.4.7.3 Framework 2: Test Case A

Here we have kept 13 legitimate nodes inside the radio frequency range of the sink node and a malicious node programmed with DDoS UDP flood attack is created and kept outside the radio range of the sink node. The node info taken during the simulation are studied and tabulated.

In this test case, one malicious node with a DIS flooding code has been added with the following algorithm :

while (i<no_of_packets)
{
i++;
send_dis();
}

The simulation time considered for the study is 600 seconds and the nodes are configured to send DIS message in every 30 seconds . The malicious node is kept out of the radio frequency range of the sink node. The simulation log file shows the packet sending time and receiving time from which we can calculate the Latency for each node.

The topology of the framework with the scenario of Node 3 being affected is shown below:

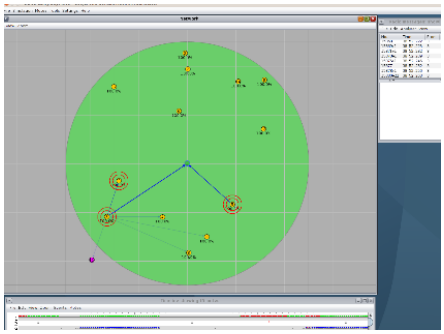
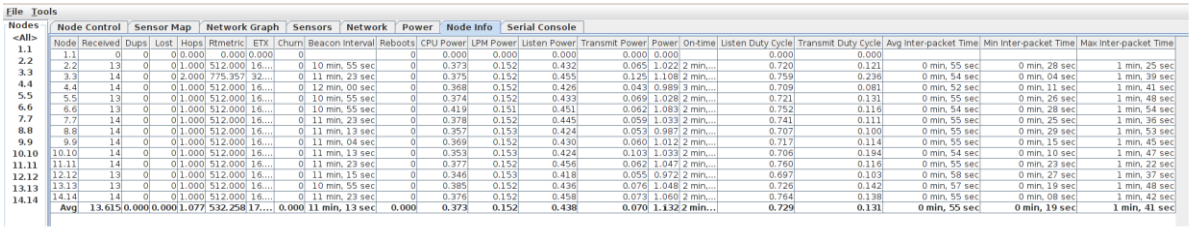


Figure 4.19: Network Topology of Framework 2: Test Case A

The Node info of the Node3 shows that the power consumed by Node 3 is higher than the other nodes and thus the average power consumed by the network also increases.



File Tools

Nodes | Node Control | Sensor Map | Network Graph | Sensors | Network | Power | Node Info | Serial Console

<All>

Node	Received	Dups	Lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power	Transmit Power	Power	On-time	Listen Duty Cycle	Transmit Duty Cycle	Avg Inter-packet Time	Min Inter-packet Time	Max Inter-packet Time
1.1	0	0	0	0.000	0.000	0.000	0		0	0.000	0.000	0.000	0.000	0.000		0.000	0.000			
2.2	13	0	0	1.000	512.000	16....	0	10 min, 55 sec	0	0.373	0.152	0.432	0.065	1.022	2 min,...	0.720	0.121	0 min, 55 sec	0 min, 28 sec	1 min, 25 sec
3.3	14	0	0	2.000	775.357	32....	0	11 min, 23 sec	0	0.375	0.152	0.455	0.125	1.108	2 min,...	0.759	0.236	0 min, 54 sec	0 min, 04 sec	1 min, 39 sec
4.4	14	0	0	1.000	512.000	16....	0	12 min, 00 sec	0	0.368	0.152	0.426	0.043	0.989	3 min,...	0.709	0.081	0 min, 52 sec	0 min, 11 sec	1 min, 41 sec
5.5	13	0	0	1.000	512.000	16....	0	10 min, 55 sec	0	0.374	0.152	0.433	0.069	1.028	2 min,...	0.721	0.131	0 min, 55 sec	0 min, 26 sec	1 min, 48 sec
6.6	13	0	0	1.000	512.000	16....	0	10 min, 55 sec	0	0.419	0.151	0.451	0.062	1.083	2 min,...	0.752	0.116	0 min, 54 sec	0 min, 28 sec	1 min, 54 sec
7.7	14	0	0	1.000	512.000	16....	0	11 min, 23 sec	0	0.378	0.152	0.445	0.059	1.033	2 min,...	0.741	0.111	0 min, 55 sec	0 min, 25 sec	1 min, 36 sec
8.8	14	0	0	1.000	512.000	16....	0	11 min, 13 sec	0	0.357	0.153	0.424	0.053	0.987	2 min,...	0.707	0.100	0 min, 55 sec	0 min, 29 sec	1 min, 53 sec
9.9	14	0	0	1.000	512.000	16....	0	11 min, 04 sec	0	0.369	0.152	0.430	0.060	1.012	2 min,...	0.717	0.114	0 min, 55 sec	0 min, 15 sec	1 min, 45 sec
10.10	14	0	0	1.000	512.000	16....	0	11 min, 13 sec	0	0.353	0.153	0.424	0.103	1.033	2 min,...	0.706	0.194	0 min, 54 sec	0 min, 10 sec	1 min, 47 sec
11.11	14	0	0	1.000	512.000	16....	0	11 min, 23 sec	0	0.377	0.152	0.456	0.062	1.047	2 min,...	0.760	0.116	0 min, 55 sec	0 min, 23 sec	1 min, 22 sec
12.12	13	0	0	1.000	512.000	16....	0	11 min, 15 sec	0	0.346	0.153	0.418	0.055	0.972	2 min,...	0.697	0.103	0 min, 58 sec	0 min, 27 sec	1 min, 37 sec
13.13	13	0	0	1.000	512.000	16....	0	10 min, 55 sec	0	0.385	0.152	0.436	0.076	1.048	2 min,...	0.726	0.142	0 min, 57 sec	0 min, 19 sec	1 min, 48 sec
14.14	14	0	0	1.000	512.000	16....	0	11 min, 23 sec	0	0.376	0.152	0.458	0.073	1.060	2 min,...	0.764	0.138	0 min, 55 sec	0 min, 08 sec	1 min, 42 sec
Avg	**13.615**	**0.000**	**0.000**	**1.077**	**532.258**	**17....**	**0.000**	**11 min, 13 sec**	**0.000**	**0.373**	**0.152**	**0.438**	**0.070**	**1.132**	**2 min...**	**0.729**	**0.131**	**0 min, 55 sec**	**0 min, 19 sec**	**1 min, 41 sec**

Figure 4.20: Node info of Framework 2 : Test Case A (13 Legitimate, 1 Sink and 1 Attack Node)

The mote output window in Cooja simulator displays the Node IDs and the messages sent/received at those nodes. Here, using the PowerTrace App we can also print the details of the power consumed by each node. By analysing this output, we can trace the flood attack and the behavioural changes happening in the victim nodes.

```
00:01.173       ID:13   nullsec CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
00:01.179       ID:3    MAC 00:12:74:03:00:03:03:03 Contiki 3.0 started. Node id is set to 3.
00:01.184       ID:13   Tentative link-local IPv6 address fe80:0000:0000:0000:0212:740d:000d:0d0d
00:01.188       ID:13   Starting 'UDP client process' 'collect common process'
00:01.188       ID:3    nullsec CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
00:01.190       ID:13   UDP client process started
00:01.195       ID:13   Client IPv6 addresses: aaaa::212:740d:d:d0d
00:01.198       ID:13   fe80::212:740d:d:d0d
00:01.199       ID:3    Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7403:0003:0303
00:01.202       ID:3    Starting 'UDP client process' 'collect common process'
00:01.203       ID:13   Created a connection with the server :: local/remote port 8775/5688
00:01.205       ID:3    UDP client process started
00:01.210       ID:3    Client IPv6 addresses: aaaa::212:7403:3:303
00:01.213       ID:3    fe80::212:7403:3:303
00:01.218       ID:3    Created a connection with the server :: local/remote port 8775/5688
00:15.111       ID:3    DATA send to 1 'Under Attack 1'
00:37.009       ID:3    DATA send to 1 'Under Attack 2'
00:42.268       ID:1    mac: turned MAC off (keeping radio on): ContikiMAC
00:47.192       ID:1    Time offset set to 3557440662
00:55.565       ID:1    I am sink!
01:11.954       ID:1    30 54282 15581 0 2313 1 1 0 22 9080 0 2887 33354 228 360 257 128 512 7 131 252 1 189
182 65535 65535 0 0 0 0
01:14.216       ID:1    30 54282 15583 0 3341 1 1 0 22 9347 0 3044 34259 239 380 257 128 512 8 131 252 1 133
```

Figure 4.21: Node 3 Attack Scenario

DDoS attacks typically flood the network with malicious traffic, forcing nodes to retransmit data to overcome packet losses. This results in increased radio transmit power consumption. The traffic causes network congestion, leading to delays and more retransmission attempts, which further increases power consumption. The RPL protocol may need to adapt to the changing network conditions caused by the attack, resulting in increased control message exchanges and higher CPU usage. Nodes with significantly higher radio transmit power are likely to be the most impacted by the DDoS attack. In this framework, we observe that Node 3 has been attacked and is under the control of the attacker node.

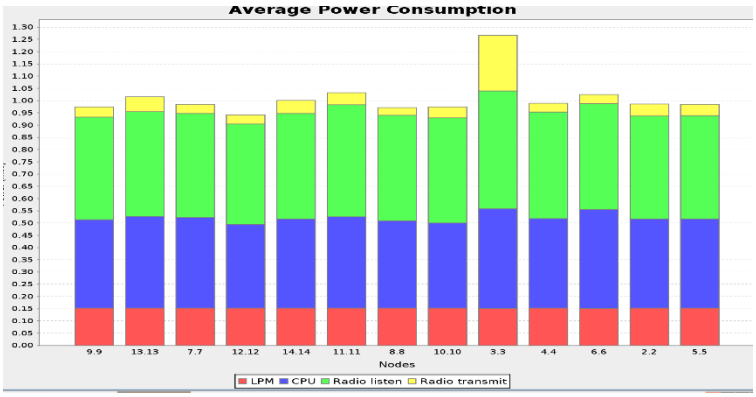


Figure 4.22: Node 3 Showing higher power consumption

The attack scenario was also studied by analysing the log files of Radio Messages generated during the simulation**.** Here we traced a high number of *Malformed Packets*. This ensures the DDoS attack experienced in the network.

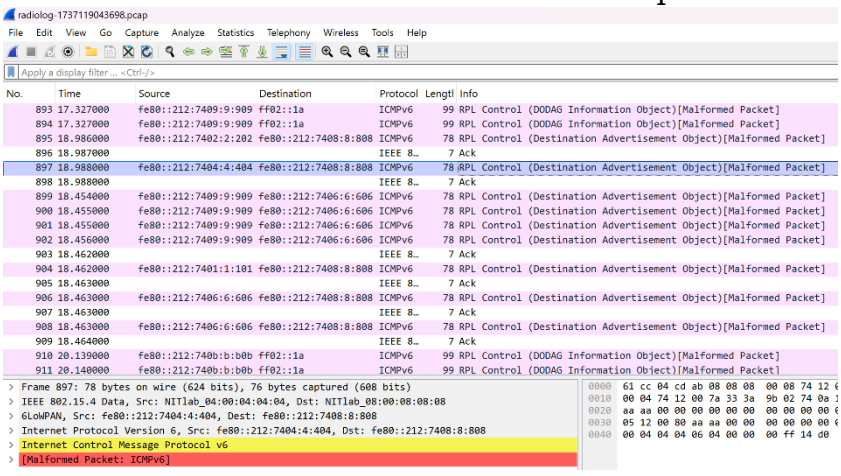

radiolog-1737119043698.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Lengtl Info
893 17.327000 fe80::212:7409:9:909 ff02::1a ICMPv6 99 RPL Control (DODAG Information Object)[Malformed Packet]
894 17.327000 fe80::212:7409:9:909 ff02::1a ICMPv6 99 RPL Control (DODAG Information Object)[Malformed Packet]
895 18.986000 fe80::212:7402:2:202 fe80::212:7408:8:808 ICMPv6 78 RPL Control (Destination Advertisement Object)[Malformed Packet]
896 18.987000 IEEE 8... 7 Ack
897 18.988000 fe80::212:7404:4:404 fe80::212:7408:8:808 ICMPv6 78 RPL Control (Destination Advertisement Object)[Malformed Packet]
898 18.988000 IEEE 8... 7 Ack
899 18.454000 fe80::212:7409:9:909 fe80::212:7406:6:606 ICMPv6 78 RPL Control (Destination Advertisement Object)[Malformed Packet]
900 18.455000 fe80::212:7409:9:909 fe80::212:7406:6:606 ICMPv6 78 RPL Control (Destination Advertisement Object)[Malformed Packet]
901 18.455000 fe80::212:7409:9:909 fe80::212:7406:6:606 ICMPv6 78 RPL Control (Destination Advertisement Object)[Malformed Packet]
902 18.456000 fe80::212:7409:9:909 fe80::212:7406:6:606 ICMPv6 78 RPL Control (Destination Advertisement Object)[Malformed Packet]
903 18.462000 IEEE 8... 7 Ack
904 18.462000 fe80::212:7401:1:101 fe80::212:7408:8:808 ICMPv6 78 RPL Control (Destination Advertisement Object)[Malformed Packet]
905 18.463000 IEEE 8... 7 Ack
906 18.463000 fe80::212:7406:6:606 fe80::212:7408:8:808 ICMPv6 78 RPL Control (Destination Advertisement Object)[Malformed Packet]
907 18.463000 IEEE 8... 7 Ack
908 18.463000 fe80::212:7406:6:606 fe80::212:7408:8:808 ICMPv6 78 RPL Control (Destination Advertisement Object)[Malformed Packet]
909 18.464000 IEEE 8... 7 Ack
910 20.139000 fe80::212:740b:b:b0b ff02::1a ICMPv6 99 RPL Control (DODAG Information Object)[Malformed Packet]
911 20.140000 fe80::212:740b:b:b0b ff02::1a ICMPv6 99 RPL Control (DODAG Information Object)[Malformed Packet]
Frame 897: 78 bytes on wire (624 bits), 76 bytes captured (608 bits)
IEEE 802.15.4 Data, Src: NITlab_04:00:04:04:04, Dst: NITlab_08:00:08:08:08
6LoWPAN, Src: fe80::212:7404:4:404, Dst: fe80::212:7408:8:808
Internet Protocol Version 6, Src: fe80::212:7404:4:404, Dst: fe80::212:7408:8:808
Internet Control Message Protocol v6
[Malformed Packet: ICMPv6]
0000 61 cc 04 cd ab 08 08 08 00 08 74 12
0010 00 04 74 12 00 7a 33 3a 9b 02 74 0a
0020 aa aa 00 00 00 00 00 00 00 00 00 00
0030 05 12 00 80 aa aa 00 00 00 00 00 00
0040 00 04 04 04 06 04 00 00 00 ff 14 d0

Figure 4.23: Radio Messages Log of Framework1 with Test Case A

The log file of radio messages generated during the simulation reveals a high volume of RPL Control messages, particularly Destination Advertisement Objects (DAOs). This indicates potential network instability, protocol overhead, or a DDoS attack. In the case of DDoS attacks, network conditions change rapidly due to packet flooding, which triggers frequent DAO exchanges as nodes attempt to re-establish routing paths. The presence of malformed packets suggests that the DDoS attack is being executed by a malicious node within the network. The high volume of RPL control messages and malformed packets destabilizes the network, leading to frequent route changes and increased latency. Processing and transmitting numerous RPL control messages can consume significant energy and processing power on the nodes, potentially impacting network performance and battery life. The increased network traffic generated by the DDoS attack, along with the resulting RPL control messages, leads to network congestion, further degrading performance and increasing packet loss rates.

Message	Number of Packets
RPL Control (DODAG Information Object)[Malformed Packet]	1256
RPL Control (Destination Advertisement Object)[Malformed P	82
RPL Control (DODAG Information Solicitation)	228
Total Packets	1609

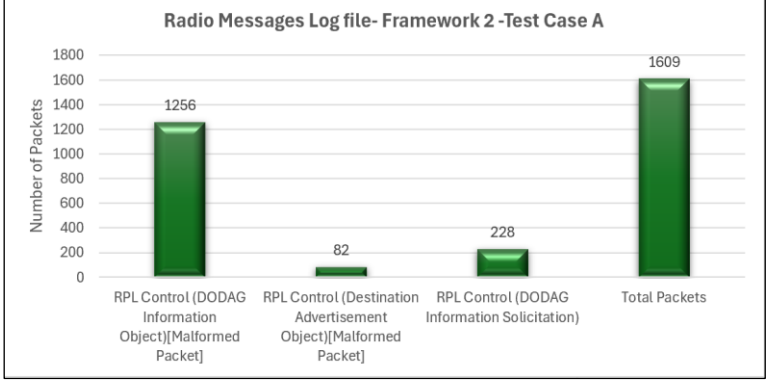


Figure 4.24: Analysis of radio messages for Framework2 -Test Case A

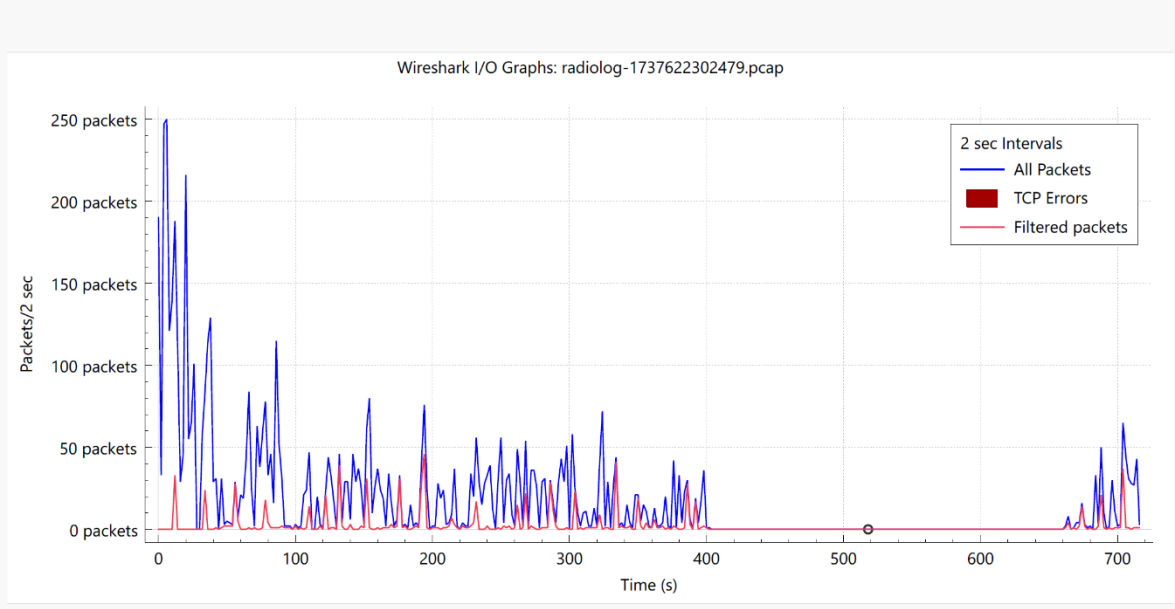


Figure 4.25: Radio log Analysis of Legitimate and Malicious Data packets

The sensor map and the network graphs recorded during the communication show the packets being delivered to node 3 takes two hops and its ETX is 32.0.

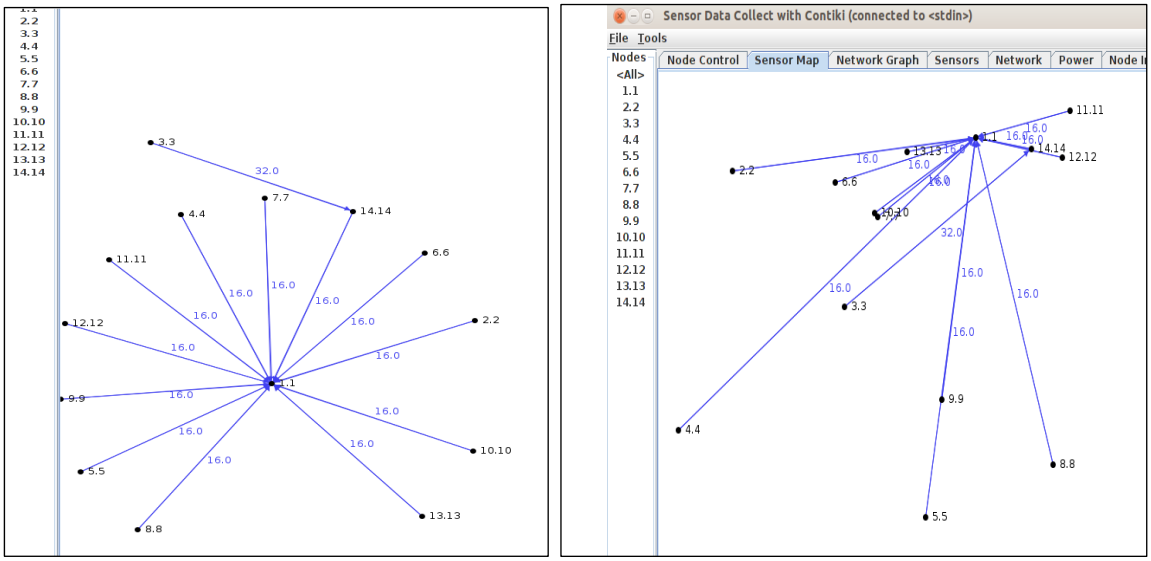


Figure 4.26: Sensor Map and Network Graph: Framework2 with Test Case B

The performance of the Framework 2 with Test Case A is studied and the following data has been recorded

Parameter	Value
PDR	96.3%
Latency	4.2
Average Power Consumed	1.132mV

Table 4.11: Performance metrics of Framework 2 with Test Case A

The average power consumed by the network remains comparatively less because the attacker node is outside the radio frequency range of all the sink node and therefor only a few sensor nodes are affected by the attack. But there is a significance drop in the packet delivery ratio and the Latency.

4.4.7.4 Framework 2 with Test Case B:

In this framework, we have included 13 legitimate nodes within the radio frequency range of the sink node, along with a malicious node programmed to execute a DDoS UDP flood attack, which is also located within the radio range of the sink node. The information gathered during the simulation is analysed and tabulated.

The topology of the framework with the scenario of Node 3 being affected is shown below:

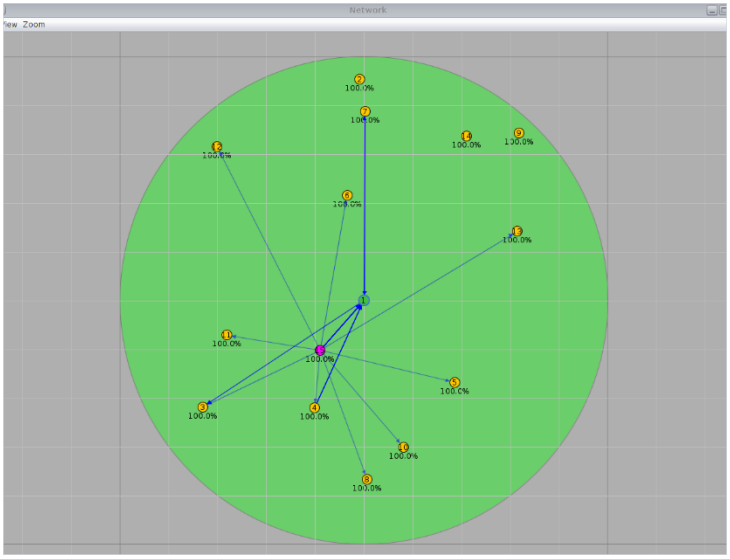
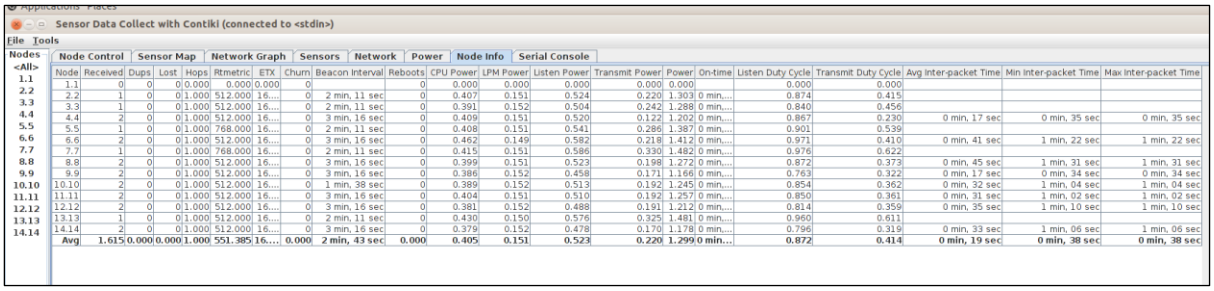


Figure 4.27: Network Topology for Framework 2 with Test Case B

In this scenario, many of the nodes are compromised by the attacker node and the network soon gets incapacitated to perform communication between the sensor nodes and the router.



Node	Received	Dups	Lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power	Transmit Power	Power	On-time	Listen Duty Cycle	Transmit Duty Cycle	Avg inter-packet Time	Min inter-packet Time	Max inter-packet Time
1.1	0	0	0	0.000	0.000	0.000	0		0	0.000	0.000	0.000	0.000	0.000		0.000	0.000			
2.2	1	0	0	1.000	512.000	16....	0	2 min, 11 sec	0	0.407	0.151	0.524	0.220	1.303	0 min,...	0.874	0.415			
3.3	1	0	0	1.000	512.000	16....	0	2 min, 11 sec	0	0.391	0.152	0.504	0.242	1.288	0 min,...	0.840	0.456			
4.4	2	0	0	1.000	512.000	16....	0	3 min, 16 sec	0	0.409	0.151	0.520	0.122	1.202	0 min,...	0.867	0.230	0 min, 17 sec	0 min, 35 sec	0 min, 35 sec
5.5	1	0	0	1.000	768.000	16....	0	2 min, 11 sec	0	0.408	0.151	0.541	0.286	1.387	0 min,...	0.901	0.539			
6.6	2	0	0	1.000	512.000	16....	0	3 min, 16 sec	0	0.462	0.149	0.582	0.218	1.412	0 min,...	0.971	0.410	0 min, 41 sec	1 min, 22 sec	1 min, 22 sec
7.7	1	0	0	1.000	768.000	16....	0	2 min, 11 sec	0	0.415	0.151	0.586	0.330	1.482	0 min,...	0.976	0.622			
8.8	2	0	0	1.000	512.000	16....	0	3 min, 16 sec	0	0.399	0.151	0.523	0.198	1.272	0 min,...	0.872	0.373	0 min, 45 sec	1 min, 31 sec	1 min, 31 sec
9.9	2	0	0	1.000	512.000	16....	0	3 min, 16 sec	0	0.386	0.152	0.458	0.171	1.166	0 min,...	0.763	0.322	0 min, 17 sec	0 min, 34 sec	0 min, 34 sec
10.10	2	0	0	1.000	512.000	16....	0	1 min, 38 sec	0	0.389	0.152	0.513	0.192	1.245	0 min,...	0.854	0.362	0 min, 32 sec	1 min, 04 sec	1 min, 04 sec
11.11	2	0	0	1.000	512.000	16....	0	3 min, 16 sec	0	0.404	0.151	0.510	0.192	1.257	0 min,...	0.850	0.361	0 min, 31 sec	1 min, 02 sec	1 min, 02 sec
12.12	2	0	0	1.000	512.000	16....	0	3 min, 16 sec	0	0.381	0.152	0.488	0.191	1.212	0 min,...	0.814	0.359	0 min, 35 sec	1 min, 10 sec	1 min, 10 sec
13.13	1	0	0	1.000	512.000	16....	0	2 min, 11 sec	0	0.430	0.150	0.576	0.325	1.481	0 min,...	0.960	0.611			
14.14	2	0	0	1.000	512.000	16....	0	3 min, 16 sec	0	0.379	0.152	0.478	0.170	1.178	0 min,...	0.796	0.319	0 min, 33 sec	1 min, 06 sec	1 min, 06 sec
Avg	**1.615**	**0.000**	**0.000**	**1.000**	**551.385**	**16....**	**0.000**	**2 min, 43 sec**	**0.000**	**0.405**	**0.151**	**0.523**	**0.220**	**1.299**	**0 min,...**	**0.872**	**0.414**	**0 min, 19 sec**	**0 min, 38 sec**	**0 min, 38 sec**

Figure 4.28 : Node info of Framework 2 with Test Case B

The sensor map and Network graph are shown below:

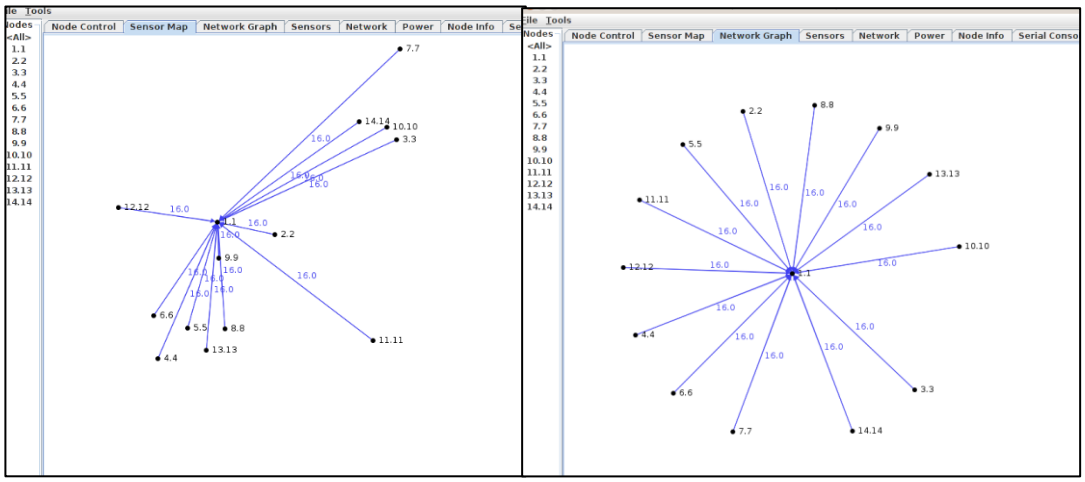


Figure 4.29 : Sensor map and network graph of Framework 2 with Test Case B

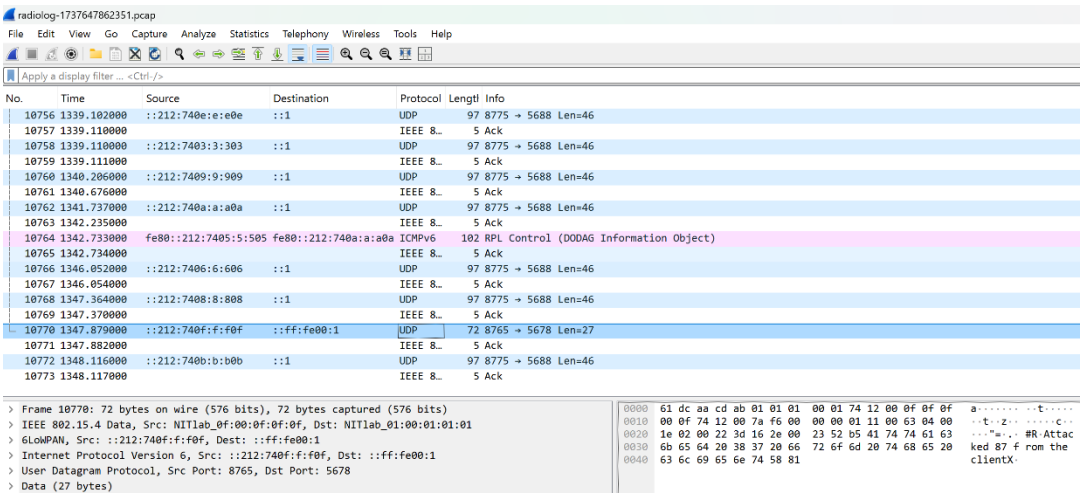


Figure 4.30: UDP Flood data packets filtered using WireShark

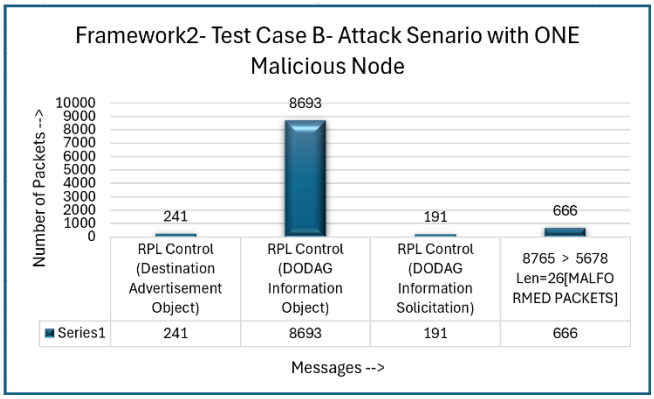


Figure 4.31: Analysis of radio messages for Framework2 -Test Case A

In test case A, only 1609 packets were transmitted over 1200s of which 1338 were malformed. In contrast, test case B in which we have placed the

malicious node within the range of the sink node and several sender nodes resulting in an alarming increase in transmitted packets(9791) thus causing the UDP flood attack. In RPL networks, the Destination Advertisement Object (DAO) indicates successful communication within the network. Here, we observed that the number of DODAG-DAOs is only 241 which shows that the entire network is under DDoS attack.

The performance of the Framework 2 with Test Case B is studied and the following data has been recorded

Parameter	Value
PDR	86.3%
Latency	5.1
Average Power Consumed	1.299mV

Table 4.12: Performance metrics of Framework 2 with Test Case B

The average power consumed by the network increases substantially because the attacker node is inside the radio frequency range of the sink node and many other nodes. and therefor only a few sensor nodes are affected by the attack. The significance drop in the packet delivery ratio and the Latency shows that the communication services are being denied among the nodes due to DDoS attack .

4.4.7.5 Framework 3: Test Case A and B

In the Framework 3 we have studied all these parameters with two test cases. In first Test Case we have taken 11 legitimate nodes, 1 sink node and 3 malicious nodes which were kept outside the radio range of the sink node. In Test Case B same scenario of test case A was considered with a major difference that the 3 malicious nodes are kept within the range of the sink node and many other sender nodes.

The performance of the Framework 3 with Test Case A and Test Case B is studied and the following data has been recorded

Parameter	Test Case A	Test Case B
PDR	80.3%	82.8%
Latency	5.5	5.2
Average Power Consumed	1.119mV	1.228mV

Table 4.13: Performance metrics of Framework3 with Test Case A and B

The average power consumed by the network in both the test cases remains almost similar because the attacker nodes are more in number and can bring the network down even though they are outside the radio frequency range of the sink node. Here many sender nodes are affected by the attack. The significance drop in the packet delivery ratio and the Latency shows that the communication services are being denied among the nodes due to DDoS attack .

4.4.7.6 Framework 4 – Test Case A and Test Case B

In this scenario, we have established a topology consisting of 11 legitimate nodes, 1 malicious node, 1 sink node and 1 IDS(Intrusion Detection System)

node programmed to detect and mitigate DDoS attacks. In Test Case A, the IDS node is positioned outside the radio range of the sink node, while in Test Case B, it is placed within the range. The analysis of various parameters during the network simulation with Test Case B demonstrated that DDoS attacks can be successfully detected and mitigated, thereby securing IoT-based smart home network at the network layer. The IDS node is coded with the following algorithm.

DDoS Detection Algorithm

Algorithm:

1. **Initialization:**
    - Initialize packet_count to 0 and last_packet_time to the current time.
    - Create a ctimer to schedule periodic packet sending.
2. **Periodic Packet Sending:**
    - Use ctimer_set() to schedule the send_packet() function to be called every PACKET_INTERVAL milliseconds.
    - send_packet() sends a UDP packet to the server.
3. **DDoS Detection:**
    - In the tcpip_handler(), when a new UDP packet is received:
        - Increment packet_count.
        - Update last_packet_time.
        - Calculate the current packet rate.
        - If the packet rate exceeds DDOS_THRESHOLD:
            - Print a "DDoS attack detected!" message.
            - Stop the packet sending timer using ctimer_stop(&send_timer).

The topology structure during the simulation along with the radio communication signals is given below:

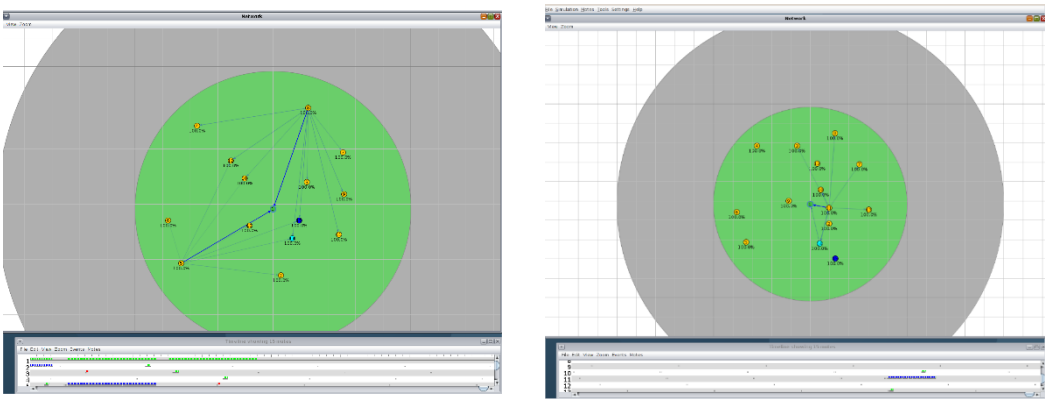
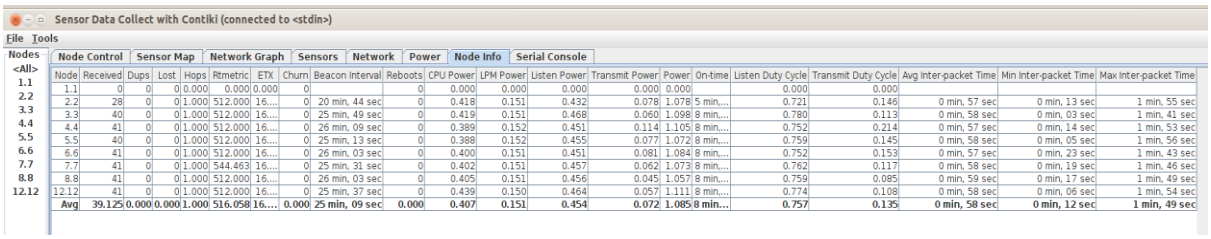


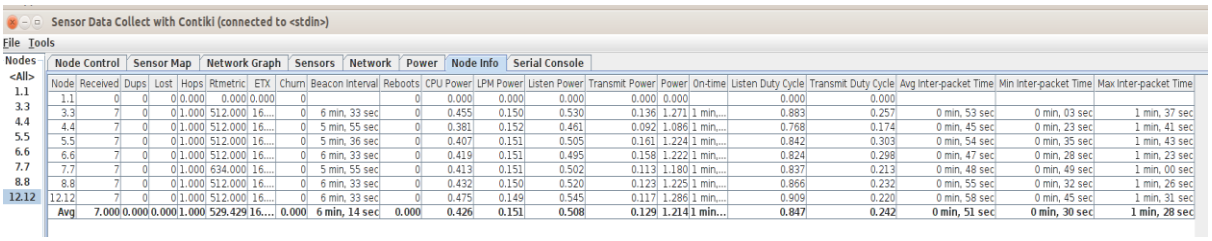
Figure 4.32: Network Topology of Framework 4 Test Case B

In this framework we observed that the network behaves erratically due to the transmission of malicious packets at the beginning, however, after some time the network stabilises. We observed and analysed several parameters during the simulation. The Node Info recorded during the first 3600seconds showed that each node was consuming higher power, higher inter-packet transmits time and higher listen and transmit Radio duty cycles. But the Node Info recorded after 25 minutes showed that the services of the network which were compromised by the attacker nodes, resumed because of the deployment of Intrusion Detection System(IDS Node).



Node	Received	Dups	Lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power	Transmit Power	Power	On-time	Listen Duty Cycle	Transmit Duty Cycle	Avg Inter-packet Time	Min Inter-packet Time	Max Inter-packet Time
1.1	0	0	0	0.000	0.000	0.000	0		0	0.000	0.000	0.000	0.000	0.000		0.000	0.000			
2.2	28	0	0	1.000	512.000	16....	0	20 min, 44 sec	0	0.418	0.151	0.432	0.078	1.078	5 min,...	0.721	0.146	0 min, 57 sec	0 min, 13 sec	1 min, 55 sec
3.3	40	0	0	1.000	512.000	16....	0	25 min, 49 sec	0	0.419	0.151	0.468	0.060	1.098	8 min,...	0.780	0.113	0 min, 58 sec	0 min, 03 sec	1 min, 41 sec
4.4	41	0	0	1.000	512.000	16....	0	26 min, 09 sec	0	0.389	0.152	0.451	0.114	1.105	8 min,...	0.752	0.214	0 min, 57 sec	0 min, 14 sec	1 min, 53 sec
5.5	40	0	0	1.000	512.000	16....	0	25 min, 13 sec	0	0.388	0.152	0.455	0.077	1.072	8 min,...	0.759	0.145	0 min, 58 sec	0 min, 05 sec	1 min, 56 sec
6.6	41	0	0	1.000	512.000	16....	0	26 min, 03 sec	0	0.400	0.151	0.451	0.081	1.084	8 min,...	0.752	0.153	0 min, 57 sec	0 min, 23 sec	1 min, 43 sec
7.7	41	0	0	1.000	544.463	16....	0	25 min, 31 sec	0	0.402	0.151	0.457	0.062	1.073	8 min,...	0.762	0.117	0 min, 58 sec	0 min, 19 sec	1 min, 46 sec
8.8	41	0	0	1.000	512.000	16....	0	26 min, 03 sec	0	0.405	0.151	0.456	0.045	1.057	8 min,...	0.759	0.085	0 min, 59 sec	0 min, 17 sec	1 min, 49 sec
12.12	41	0	0	1.000	512.000	16....	0	25 min, 37 sec	0	0.439	0.150	0.464	0.057	1.111	8 min,...	0.774	0.108	0 min, 58 sec	0 min, 06 sec	1 min, 54 sec
Avg	**39.125**	**0.000**	**0.000**	**1.000**	**516.058**	**16....**	**0.000**	**25 min, 09 sec**	**0.000**	**0.407**	**0.151**	**0.454**	**0.072**	**1.085**	**8 min...**	**0.757**	**0.135**	**0 min, 58 sec**	**0 min, 12 sec**	**1 min, 49 sec**

Figure 4.33: Node Info of Framework4 Test Case B (First 10 Minutes)



Node	Received	Dups	Lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power	Transmit Power	Power	On-time	Listen Duty Cycle	Transmit Duty Cycle	Avg Inter-packet Time	Min Inter-packet Time	Max Inter-packet Time
1.1	0	0	0	0.000	0.000	0.000	0		0	0.000	0.000	0.000	0.000	0.000		0.000	0.000			
3.3	7	0	0	1.000	512.000	16....	0	6 min, 33 sec	0	0.455	0.150	0.530	0.136	1.271	1 min,...	0.883	0.257	0 min, 53 sec	0 min, 03 sec	1 min, 37 sec
4.4	7	0	0	1.000	512.000	16....	0	5 min, 55 sec	0	0.381	0.152	0.461	0.092	1.086	1 min,...	0.768	0.174	0 min, 45 sec	0 min, 23 sec	1 min, 41 sec
5.5	7	0	0	1.000	512.000	16....	0	5 min, 36 sec	0	0.407	0.151	0.505	0.161	1.224	1 min,...	0.842	0.303	0 min, 54 sec	0 min, 35 sec	1 min, 43 sec
6.6	7	0	0	1.000	512.000	16....	0	6 min, 33 sec	0	0.419	0.151	0.495	0.158	1.222	1 min,...	0.824	0.298	0 min, 47 sec	0 min, 28 sec	1 min, 23 sec
7.7	7	0	0	1.000	634.000	16....	0	5 min, 55 sec	0	0.413	0.151	0.502	0.113	1.180	1 min,...	0.837	0.213	0 min, 48 sec	0 min, 49 sec	1 min, 00 sec
8.8	7	0	0	1.000	512.000	16....	0	6 min, 33 sec	0	0.432	0.150	0.520	0.123	1.225	1 min,...	0.866	0.232	0 min, 55 sec	0 min, 32 sec	1 min, 26 sec
12.12	7	0	0	1.000	512.000	16....	0	6 min, 33 sec	0	0.475	0.149	0.545	0.117	1.286	1 min,...	0.909	0.220	0 min, 58 sec	0 min, 45 sec	1 min, 31 sec
Avg	**7.000**	**0.000**	**0.000**	**1.000**	**529.429**	**16....**	**0.000**	**6 min, 14 sec**	**0.000**	**0.426**	**0.151**	**0.508**	**0.129**	**1.214**	**1 min...**	**0.847**	**0.242**	**0 min, 51 sec**	**0 min, 30 sec**	**1 min, 28 sec**

Figure 4.34: Node Info of Framework4 Test Case B after 25 Minutes

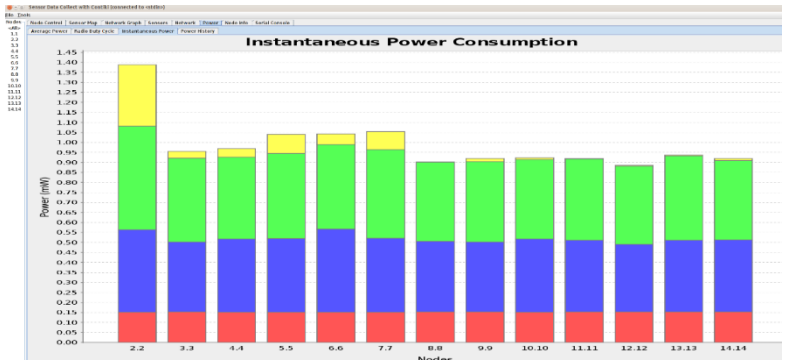


Figure 4.35: Average Power consumption Framework 4 Test Case B – First 10 minutes duration

Here because of the DDoS attack, the number of packets being transmitted increases rapidly in the beginning and some of the nodes stop transmitting radio signals and their CPU and Radio receive power consumption was high for some period. At the end of 25 minutes of simulation, the network stabilizes, and the average power consumptions becomes regulated. We can observe here that In Graph 2, the combined height of the "Radio Listen" and "Radio Transmit" bars is noticeably lower compared to Graph 1. This indicates that the IDS is effectively filtering out a significant portion of the malicious traffic, reducing the overall radio activity in the network. The overall height of

the bars in Graph 2 is generally lower than in Graph 1. This indicates a significant reduction in average power consumption across the network when the IDS is enabled. The IDS is effectively mitigating the impact of the DDoS attack by filtering out malicious traffic.

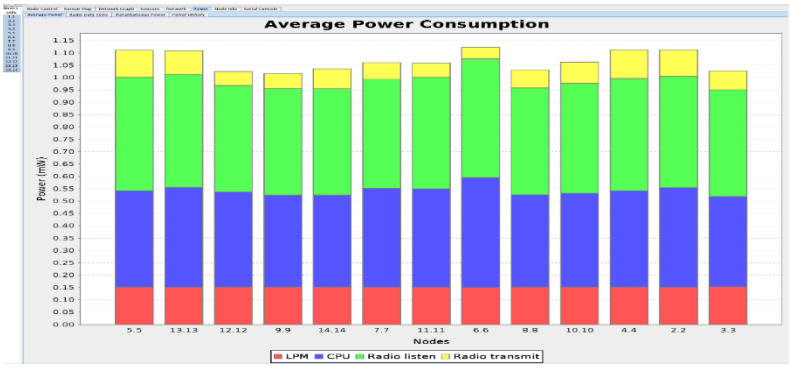


Figure 4.36: Average Power consumption Framework 4 Test Case B – After 25 minutes

The network's behaviour is also studied using the parameters like Mote Outputs and Radio Listener logs. The Mote outputs describe the activities of the network. Here in this framework Node 14 is the attacker node and during the initial stages of the simulation, Node 14 sends attack messages to the router.

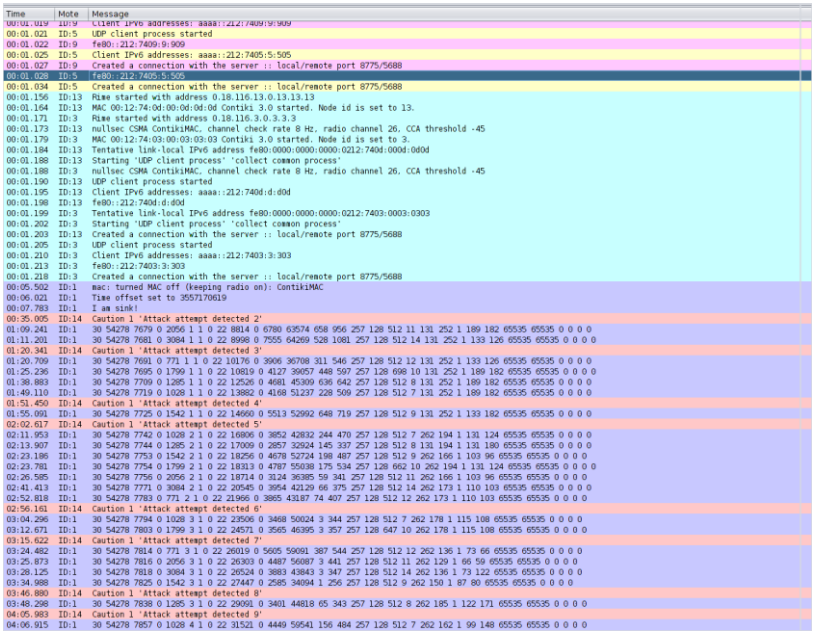


Figure 4.37: Mote outputs of Framework 4 Test Case B – First 10 minutes.

04 00.763 ID:15 Tentative link-local IPv6 address fe80:0000:0000:0000:0212:740f:000f:0f0f
04 00.766 ID:14 Created a connection with the server :: local/remote port 8765/5678
04:00.766 ID:15 Starting 'UDP client process' 'collect common process'
04:00.769 ID:15 UDP client process started
04:00.774 ID:15 Client IPv6 addresses: aaaa::212:740f:f:f0f
04:00.776 ID:15 fe80::212:740f:f:f0f
04:00.780 ID:15 Successfully joined RPL DAG
C5:00.155 ID:15 Intrusion Detection System Enabled :: local/remote port 8775/5688

Figure 4.38: Mote outputs of Framework 4 Test Case B – IDS enabled.

Here the Node 15(IDS) starts detecting the attack attempts by Node 14(Attacker) and after this the network stabilizes and the number of packets being transmitted gets reduced. We can observe the Mote Output "Intrusion

Detection System Enabled: local/remote port 8775/5688:" This line indicates that the IDS has been successfully enabled on the mote 15. It specifies the local and remote ports used for IDS communication.

```
01:38.883  ID:1   30 54278 7709 0 1285 1 1 0 22 12526 0 4681 45309 636 642 257 128 512 8 131 252 1 189 182 65535 65535 0 0 0 0
01:49.110  ID:1   30 54278 7719 0 1028 1 1 0 22 13882 0 4168 51237 228 509 257 128 512 7 131 252 1 189 182 65535 65535 0 0 0 0
01:51.450  ID:15  Caution 1 'Attack attempt detected 4'
01:55.091  ID:1   30 54278 7725 0 1542 1 1 0 22 14660 0 5513 52992 648 719 257 128 512 9 131 252 1 133 182 65535 65535 0 0 0 0
02:02.617  ID:15  Caution 1 'Attack attempt detected 5'
02:11.953  ID:1   30 54278 7742 0 1028 2 1 0 22 16806 0 3852 42832 244 470 257 128 512 7 262 194 1 131 124 65535 65535 0 0 0 0
02:13.907  ID:1   30 54278 7744 0 1285 2 1 0 22 17009 0 2857 32924 145 337 257 128 512 8 131 194 1 131 180 65535 65535 0 0 0 0
02:23.186  ID:1   30 54278 7753 0 1542 2 1 0 22 18256 0 4678 52724 198 487 257 128 512 9 262 166 1 103 96 65535 65535 0 0 0 0
02:23.781  ID:1   30 54278 7754 0 1799 2 1 0 22 18313 0 4787 55038 175 534 257 128 662 10 262 194 1 131 124 65535 65535 0 0 0 0
02:26.585  ID:1   30 54278 7756 0 2056 2 1 0 22 18714 0 3124 36385 59 341 257 128 512 11 262 166 1 103 96 65535 65535 0 0 0 0
02:41.413  ID:1   30 54278 7771 0 3084 2 1 0 22 20545 0 3954 42129 66 375 257 128 512 14 262 173 1 110 103 65535 65535 0 0 0 0
02:52.818  ID:1   30 54278 7783 0 771 2 1 0 22 21966 0 3865 43187 74 407 257 128 512 12 262 173 1 110 103 65535 65535 0 0 0 0
02:56.161  ID:15  Caution 1 'Attack attempt detected 6'
03:04.296  ID:1   30 54278 7794 0 1028 3 1 0 22 23506 0 3468 50024 3 344 257 128 512 7 262 178 1 115 108 65535 65535 0 0 0 0
03:12.671  ID:1   30 54278 7803 0 1799 3 1 0 22 24571 0 3565 46395 3 357 257 128 647 10 262 178 1 115 108 65535 65535 0 0 0 0
03:15.622  ID:15  Caution 1 'Attack attempt detected 7'
03:24.482  ID:1   30 54278 7814 0 771 3 1 0 22 26019 0 5605 59091 387 544 257 128 512 12 262 136 1 73 66 65535 65535 0 0 0 0
03:25.873  ID:1   30 54278 7816 0 2056 3 1 0 22 26303 0 4487 56087 3 441 257 128 512 11 262 129 1 66 59 65535 65535 0 0 0 0
03:28.125  ID:1   30 54278 7818 0 3084 3 1 0 22 26524 0 3883 43843 3 347 257 128 512 14 262 136 1 73 122 65535 65535 0 0 0 0
03:34.988  ID:1   30 54278 7825 0 1542 3 1 0 22 27447 0 2585 34094 1 256 257 128 512 9 262 150 1 87 80 65535 65535 0 0 0 0
03:46.880  ID:15  Caution 1 'Attack attempt detected 8'
03:48.298  ID:1   30 54278 7838 0 1285 3 1 0 22 29091 0 3401 44818 65 343 257 128 512 8 262 185 1 122 171 65535 65535 0 0 0 0
04:05.983  ID:15  Caution 1 'Attack attempt detected 9'
04:06.915  ID:1   30 54278 7857 0 1028 4 1 0 22 31521 0 4449 59541 156 484 257 128 512 7 262 162 1 99 148 65535 65535 0 0 0 0
```

Figure 4.39: Attack detection message by IDS Node

The provided output demonstrates the IDS successfully detecting and logging multiple attack attempts and the IDS is effectively mitigating the attack and maintaining the stability of the RPL network. At the end of 25 minutes of simulations we found that no more attacks were performed by the attacker node. This is analysed using the Radio messages log file and comparing the counts of DIO,DIS and DAO control messages in the RPL network.

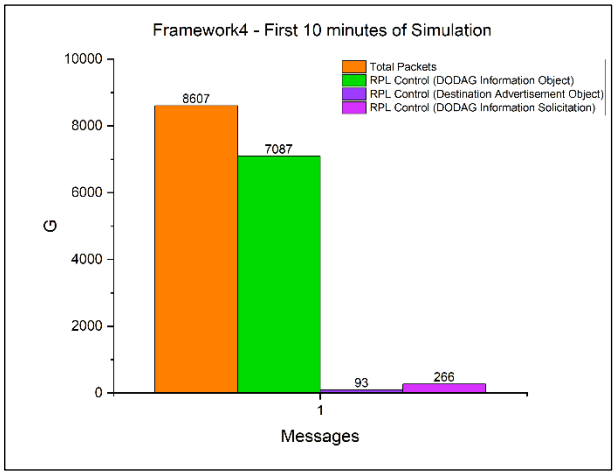


Figure 4.40 (a) Framework 4 – Radio Messages Log Analysis During first 10 minutes of simulation.

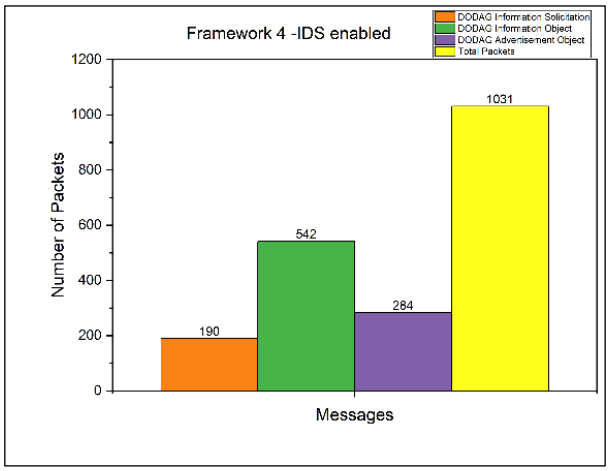


Figure 4.40 (b) Framework 4 – Radio Messages Log Analysis During after 25 minutes of simulation.

The log file of radio messages transmitted during this simulation showed a clear scenario of DDoS attack during the first 600 seconds of simulation. In this test case total 8607 packets were transmitted over first 2 minutes of simulation among which 7087 DIO were found. But number of the Destination Advertisement Object (93) clearly indicates the network has been compromised by the flooding attack.

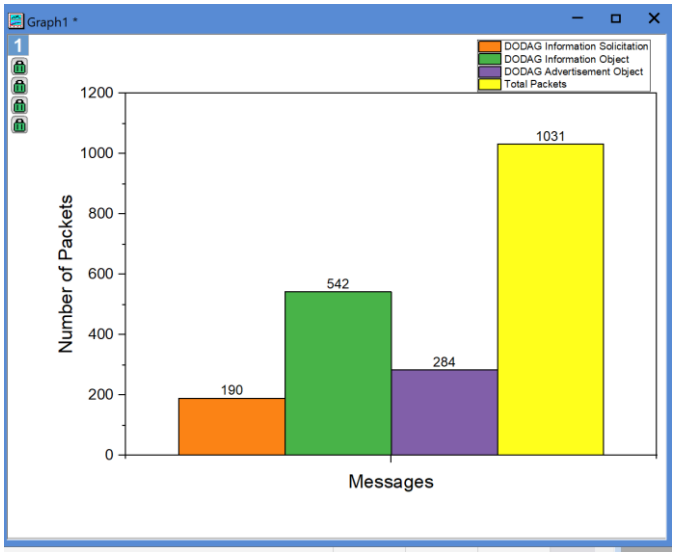


Figure 4.41: Radio messages Log Analysis - After IDS gets enabled

After enabling IDS, the number of packets being transmitted stabilizes and all nodes start communicating in the network.

The performance of the Framework 4 with Test Case A and Test Case B after enabling IDS is studied and the following data has been recorded

Parameter	Test Case A	Test Case B
PDR	90.3%	92.8%
Latency	4.5	4.2
Average Power Consumed	1.105mV	1.085mV

Table 4.14: Performance metrics of Framework 4 with Test Cases A & B

The average power consumed by the network now improved in both the test cases because the attacker nodes do not send any message floods. The packet delivery ratio and the Latency also show significant improvement that the communication services are being restored among the nodes which was disturbed due to DDoS attack.

4.4.8 CONCLUSION

The simulations performed in Cooja simulator to implement ARIDS clearly shows that the DDoS attack can be mitigated with AtRouter Intrusion Detection System. The overall behaviour of the network is as shown below.

Frame work	Test Case	PDR	Latency	Average Power Consumed
1	A (Normal)	99.3	3.5	1.097
1	B (Normal)	99.8	3.2	1.087
2	A (Single attacker)	96.3	4.2	1.132
2	B (Single attacker)	86.3	5.1	1.299
3	A (3 attack nodes)	80.3	5.5	1.119
3	B (3 attack nodes)	82.8	5.2	1.228
4	A (IDS- outside)	90.3	4.5	1.105
4	B (IDS - inside)	96.8	4.2	1.085

Table 4.15: Comparison of various performance metrics across all Frameworks

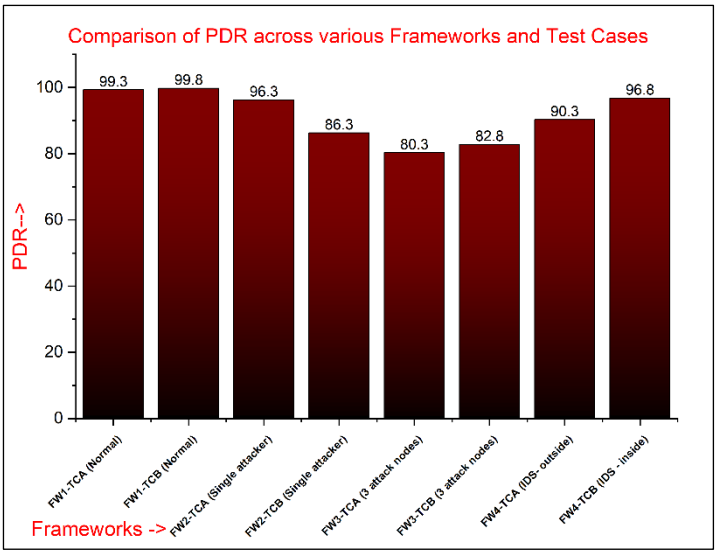


Figure 4.42: Comparison of Packet Delivery Ratio across various Frameworks and test cases

The Framework 1 is the baseline scenario with no attacker nodes or specific security measures. Here in the test case B, PDR is very high, indicating a reliable network under normal conditions but in test case A a small variation can be found because 50% of the sender nodes are outside the radio frequency range of the server node.

In the Framework 2, test case B shows more packet drops due to DDoS attack by the single attacker node which is placed inside the radio range of sink node where as in test case A, though the network is under attack many of the nodes which do not come under the radio range of the attacker node function properly thus maintain a high PDR compared to that of test case B.

In Framework 3, the Packet Delivery Ratio (PDR) drastically decreases due to a severe DDoS attack involving three nodes. In test case B, the PDR slightly improves because the compromised nodes, which are actually legitimate nodes, begin sending malformed packets to the sink node.

In the Framework 4, PDR regains the value which was obtained during normal scenario without any attacker nodes present. This clearly shows the efficient implementation of IDS which can effectively filter out malicious traffic before it reaches the core network. The difference between the PDRs in test case A and B in Framework 4 is negligible which indicates that the IDS is effective regardless of its placement, although the internal placement might offer additional benefits like more granular monitoring.

In the graph Figure 4.43, we have compared the Latency of the network across various frameworks and test cases. The increase in Latency shows the inefficiency of the network. The graph clearly shows higher latency at framework 3, indicating that DDoS attack reduces the efficiency of the network services. The latency recovers when IDS is enabled at Framework 4. The latency is lower in all B test cases because the nodes are positioned within the radio range of the sink node. The distance between the sender nodes and the sink node contributes to delay as the sender nodes may use two or more hops to reach the data packets to its destination.

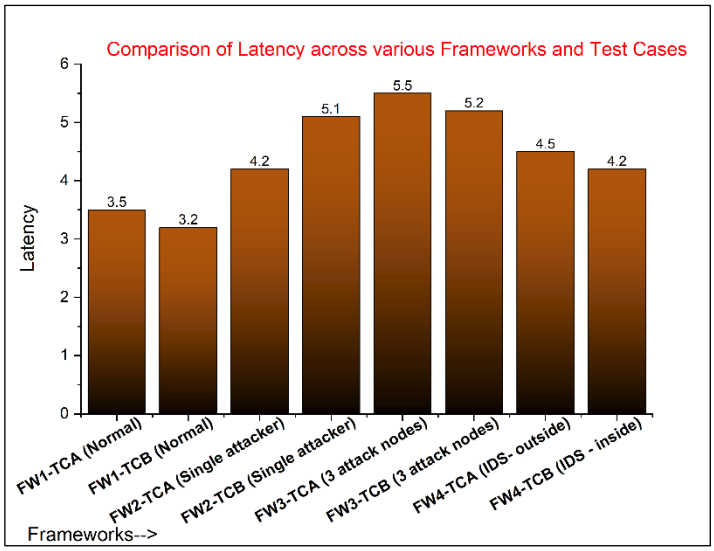


Figure 4.43: Comparison of Latency across various Frameworks and test cases.

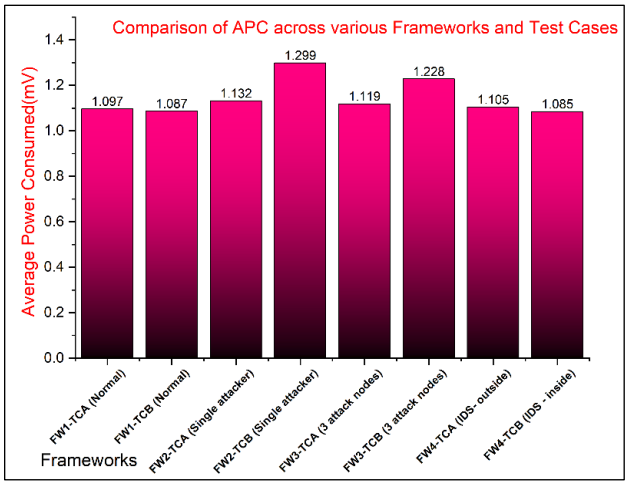


Figure 4.44: Comparison of Average Power Consumption (APC) across Frameworks and test cases.

The comparative study of Average Power Consumption across different frameworks and test cases in this project shows that sender nodes consume more energy in test case A than in test case B due to the extended distance between the sender nodes and the sink node in test case A. Thus, test case B introduces minor optimization in terms of power consumption, with or without an attack. The attack by 3 malicious nodes contribute to higher CPU and energy utilization. The enabling of IDS stabilises the network to its normal behaviour and leads to substantial decrease in power consumption. Thus, we conclude that it is crucial to consider the energy constraints of IoT devices when designing security solutions.

5 CONCLUSION

We have successfully designed and implemented a multi-layered security approach for IoT-based smart home networks to address their inherent vulnerabilities. The project focussed on lightweight solutions acknowledging the resource constraints of IoT-based networks. A Lightweight Intrusion

Detection System (LWIDS) was developed, utilizing lightweight cryptographic algorithms such as IDEA, HIGHT, and PRESENT to secure communication between sensors and gateways, at the application layer. Additionally, a DDoS attack prevention and mitigation mechanism, AtRouterIDS (ARIDS), was implemented at the network layer to protect against malicious attacks utilizing the RPL protocol. The combined effect of LWIDS encrypting sensor-gateway communication and ARIDS detecting and preventing DDoS attacks at the network layer collectively enhances the overall security of the smart home network without attenuating the overall performance of the network.

6 FUTURE SCOPE

To overcome the resource constraints of the IoT-based smart home networks, in future we have planned to design and develop security models using Machine Learning and Deep Learning algorithms and implement them in the cloud.

PUBLICATIONS

1. N. Suparna and D. H. Manjaiah, "Implementation and Performance Analysis of IDEA in IoT-Based Smart Home Networks," *International Conference on Artificial Intelligence and Data Engineering, AIDE 2022*, pp. 212–216, 2022, doi: 10.1109/AIDE57180.2022.10060851.
2. N. Suparna and D. H. Manjaiah, "A Comprehensive Study of Datasets for Research Works in IoT Networks" *International Journal of Science and Research Archive*
3. N. Suparna and D. H. Manjaiah, "A Comprehensive Survey of Operating Systems for Smart Home Networks Based on IOT - *International Journal of Engineering Research & Technology (IJERT),* ISSN: 2278-0181, Vol. 13 Issue 10, October 2024
4. N. Suparna and D. H. Manjaiah, "Detection of DDOS Attacks in IoT Based Smart Home Networks using ML Algorithms" – *Springer LNNS -Data Analytics and Learning: Proceedings of DAL 2024: Editors: D. S. Guru, N. Vinay Kumar, Mohammed Javed* * ACCEPTED FOR PUBLICATION

REFERENCES

[1] M. Schiefer, "Smart Home Definition and Security Threats," in *2015 Ninth International Conference on IT Security Incident Management & IT Forensics*, Magdeburg, Germany: IEEE, May 2015, pp. 114–118. doi: 10.1109/IMF.2015.17.

[2] M. Chan, D. Estève, C. Escriba, and E. Campo, "A review of smart homes-Present state and future challenges," *Computer Methods and Programs in Biomedicine*, vol. 91, no. 1, pp. 55–81, 2008, doi: 10.1016/j.cmpb.2008.02.001.

[3] J. P. Gabhane, M. S. Thakare, and M. M. Craig, "Smart Homes System Using Internet-of-Things : Issues , Solutions and Recent Research Directions," 2017.

[4] S. B. Venkata *et al.*, "Security Analysis in Wireless Sensor Networks," *IEEE*, vol. 2014, no. 13, pp. 1292–1297, 2017, doi: 10.1109/JSEN.2017.2705045.

[5] S. Alam and D. De, "Analysis of Security Threats in Wireless Sensor Network," *International Journal of Wireless & Mobile Networks*, vol. 6, no. 2, pp. 35–46, 2014, doi: 10.5121/ijwmn.2014.6204.

[6] G. Sharma, S. Vidalis, N. Anand, C. Menon, and S. Kumar, "A survey on layer-wise security attacks in iot: Attacks, countermeasures, and open-issues," *Electronics (Switzerland)*, vol. 10, no. 19, pp. 1–23, 2021, doi: 10.3390/electronics10192365.

[7] H. Sajjad, "Evaluating Security Threats for each Layers of IoT System Evaluating Security Threats for each Layers of IoT System," no. October 2019, pp. 0–6, 2020.

[8] J. Bugeja, A. Jacobsson, and P. Davidsson, "On Privacy and Security Challenges in Smart Connected Homes," *IEEE Computer Society*, 2016, doi: 10.1109/EISIC.2016.21.

[9] A. Gai, S. Azam, B. Shanmugam, M. Jonkman, and F. De Boer, “Categorisation of security threats for smart home appliances,” *2018 International Conference on Computer Communication and Informatics, ICCCI 2018*, pp. 1–5, 2018, doi: 10.1109/ICCCI.2018.8441213.
[10] I. Alsmadi and F. Mira, “IoT security threats analysis based on components, layers and devices,” *American Journal of Science & Engineering*, vol. 1, no. 1, pp. 1–10, 2020, doi: 10.15864/ajse.1101.
[11] A. E. Coboi, T. Que Son, M. T. Nguyen, A. Ernesto Coboi, T. A. Tran, and S. Q. Tran, “Security Problems in Smart Homes,” *ICSES Transactions on Computer Networks and Communications*, no. Y, 2021.
[12] S. Yoon, H. Park, and H. S. Yoo, “Security issues on smarthome in IoT environment,” *Lecture Notes in Electrical Engineering*, vol. 330, pp. 691–696, 2015, doi: 10.1007/978-3-662-45402-2_97.
[13] X. Lai and J. L. Massey, “A proposal for a new block encryption standard,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 473 LNCS, pp. 389–404, 1991, doi: 10.1007/3-540-46877-3_35.
[14] N. Suparna and D. H. Manjaiah, “Implementation and Performance Analysis of IDEA in IoT-Based Smart Home Networks,” *International Conference on Artificial Intelligence and Data Engineering, AIDE 2022*, pp. 212–216, 2022, doi: 10.1109/AIDE57180.2022.10060851.
[15] A. Curigert, H. Bonnenbergt, R. Zimmermanmt, N. Felbert, H. Kaeslinz, and W. Fichtnert, “VINCI: VLSI IMPLEMENTATION OF THE NEW SECRET-KEY BLOCK CIPHER: IDEA,” *IEEE*, pp. 3–6, 1993, doi: 0-7803-0826-3/93.
[16] R. Ranjan and I. Poonguzhali, “VLSI Implementation of IDEA Encryption Algorithm,” 1992.
[17] A. Buldas, “A VLSI implementation of RSA and IDEA encryption”.
[18] M. P. Leong, O. Y. H. Cheung, K. H. Tsoi, and P. H. W. Leong, “A bit-serial implementation of the International Data Encryption Algorithm IDEA,” *IEEE Symposium on FPGAs for Custom Computing Machines, Proceedings*, vol. 2000-Janua, pp. 122–131, 2000, doi: 10.1109/FPGA.2000.903399.
[19] H. Bonnenberg, A. Curiger, N. Felber, H. Kaeslin, and X. Lai, “VLSI implementation of a new block cipher,” *IEEE International Conference on Computer Design - VLSI in Computers and Processors*, pp. 510–513, 1991, doi: 10.1109/iccd.1991.139960.
[20] S. Wolter, M. Holger, A. Schubert, and R. Laur, “On the VLSI Implementation of the International Data Encryption Algorithm IDEA,” *IEEE*, pp. 397–400, 1995, doi: 0-7803-2570-2/95.
[21] U. Tandale, B. Momin, and D. P. Seetharam, “An Empirical Study of Application Layer Protocols for IoT,” *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 2447–2451, 2017.
[22] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, “Cross-Level Sensor Network Simulation with COOJA,” pp. 641–648, 2006.
[23] “Z1 Mote.” Accessed: Oct. 09, 2022. [Online]. Available: https://github.com/contiki-os/contiki/wiki/Zolertia-z1-motes
[24] D. Irwin, P. Liu, S. R. Chaudhry, M. Collier, and X. Wang, “A Performance Comparison of the PRESENT Lightweight Cryptography Algorithm on Different Hardware Platforms,” *2018 29th Irish Signals and Systems Conference (ISSC)*, pp. 1–5, 2018.
[25] E. Jacinto and H. Montiel, “Implementation of the cryptographic algorithm ‘ Present ’ in different microcontroller type embedded software platforms,” 2017.
[26] A. Bogdanov *et al.*, “PRESENT: An Ultra-Lightweight Block Cipher,” in *Cryptographic Hardware and Embedded Systems - CHES 2007*, P. Paillier and I. Verbauwhede, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466.
[27] A. Bogdanov, Ed., “Lightweight Cryptography for Security and Privacy - 5th International Workshop, LightSec 2016, Aksaray, Turkey, September 21-22, 2016, Revised Selected Papers,” in Lecture Notes in Computer Science, vol. 10098. Springer, 2017. doi: 10.1007/978-3-319-55714-4.
[28] C. Paar and J. Pelzl, *Understanding Cryptography*.

[29] M. Sbeiti, M. Silbermann, A. Poschmann, and C. Paar, “Design space exploration of present implementations for FPGAS,” in *2009 5th Southern Conference on Programmable Logic (SPL)*, São Carlos, Brazil: IEEE, Apr. 2009, pp. 141–145. doi: 10.1109/SPL.2009.4914893.
[30] P. V. J. Hengameh Delfan Azari, “AN EFFICIENT IMPLEMENTATION OF PRESENT CIPHER MODEL WITH 80 BIT AND 128 BIT KEY OVER FPGA BASED HARDWARE ARCHITECTURE,” vol. 119, no. 14, pp. 1825–1832, 2018.
[31] Kiran Kumar V G, Asmita Poojary, Shantharama Rai C, and Nagesh H R, “Implementation of Lightweight Cryptographic Algorithms in FPGA,” *2017 International Conference on Circuits, Controls, and Communications (CCUBE)*, pp. 232–235, 2017.
[32] P. Yalla and J. Kaps, “Lightweight Cryptography for FPGAs,” in *IEEE*, IEEE, 2009, pp. 225–230. doi: 10.1109/ReConFigure.2009.54.
[33] E. Gomez, U. Distrital, F. José, and C. Hernández, “Performance Evaluation of the Present Cryptographic Algorithm over FPGA,” vol. 10, no. 12, pp. 555–567, 2017.
[34] D. Hong *et al.*, “HIGHT: A New Block Cipher Suitable for Low-Resource Device,” in *Cryptographic Hardware and Embedded Systems - CHES 2006*, L. Goubin and M. Matsui, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 46–59.
[35] J. Aguilar, S. Sierra, and E. Jacinto, “Implementation of ‘HIGHT’ encryption algorithm on microcontroller,” *CHILECON 2015 - 2015 IEEE Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies, Proceedings of IEEE Chilecon 2015*, pp. 937–942, 2016, doi: 10.1109/Chilecon.2015.7404685.
[36] B. Kim, J. Cho, B. Choi, J. Park, and H. Seo, “Compact Implementations of HIGHT Block Cipher on IoT Platforms,” *Security and Communication Networks*, vol. 2019, 2019, doi: 10.1155/2019/5323578.
[37] B. J. Mohd, T. Hayajneh, Z. A. Khalaf, and K. M. Ahmad Yousef, “Modeling and optimization of the lightweight HIGHT block cipher design with FPGA implementation,” *Security and Communication Networks*, vol. 9, no. 13, pp. 2200–2216, 2016, doi: 10.1002/sec.1479.
[38] M. M. Salim, S. Rathore, and J. H. Park, *Distributed denial of service attacks and its defenses in IoT: a survey*, vol. 76, no. 7. Springer US, 2020. doi: 10.1007/s11227-019-02945-z.
[39] T. Mehmood, “COOJA Network Simulator: Exploring the Infinite Possible Ways to Compute the Performance Metrics of IOT Based Smart Devices to Understand the Working of IOT Based Compression & Routing Protocols,” *ArXiv*, 2017, doi: https://doi.org/10.48550/arXiv.1712.08303.
[40] P. Spadaccino and F. Cuomo, “INTRUSION DETECTION SYSTEMS FOR IOT: OPPORTUNITIES AND CHALLENGES OFFERED BY EDGE COMPUTING AND MACHINE LEARNING”.
[41] S. Cakir, S. Toklu, and N. Yalcin, “RPL Attack Detection and Prevention in the Internet of Things Networks Using a GRU Based Deep Learning,” *IEEE Access*, vol. 8, pp. 183678–183689, 2020, doi: 10.1109/ACCESS.2020.3029191.
[42] V. R. Rajasekar and S. Rajkumar, “Analysis of Blackhole Attack in RPL-based 6LoWPAN Network: A Case Study,” in *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Dubai, United Arab Emirates: IEEE, Nov. 2021, pp. 1–6. doi: 10.1109/ICECS53924.2021.9665623.

Research Scholar

Suparna N
Research scholar,
Dept. of Compute science,
Mangalore University.

Research Supervisor

Dr. Manjaiah D H
Senior Professor & Chairman
Dept. of Compute science,
Mangalore University.